Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase-Space Knowledge

by

Feng Zhao

Submitted to the Department of Electrical Engineering and Computer Science
on August 19, 1992 in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Science
at Massachusetts Institute of Technology

Abstract

This thesis presents a novel design methodology for the synthesis of automatic controllers, together with a computational environment—the Control Engineer's Workbench—integrating a suite of programs that automatically analyze and design controllers for high-performance, global control of nonlinear systems. This work demonstrates that difficult control synthesis tasks can be automated, using programs that actively exploit and efficiently represent knowledge of nonlinear dynamics and phase space and effectively use the representation to guide and perform the control design. The Control Engineer's Workbench combines powerful numerical and symbolic computations with spatial reasoning techniques. The two major programs in the Workbench—Phase Space Navigator and MAPS—work together to model and reason about the phase-space geometry and topology of a given system, to plan global control reference trajectories, and to navigate the system along the planned trajectories. They use a novel technique of "flow pipes" to group infinite numbers of distinct behaviors into a manageable discrete set that becomes the basis for establishing the reference trajectories.

As a demonstration of this approach, I exhibit the automatic design of a nonlinear controller for a magnetic levitation system. The control system synthesized by the Workbench can stabilize a maglev vehicle with large initial displacements from an equilibrium and outperform the classical linear feedback design for the same system by a factor of 20.

Keywords. Artificial intelligence, scientific computing, control system design, numeric/symbolic processing, qualitative reasoning, geometric modeling, nonlinear dynamics.

Thesis Supervisors: Harold Abelson and Gerald Jay Sussman

To my parents

Acknowledgments

Many people have contributed to my intellectual growth and the preparation of this thesis. I am especially grateful to the following people.

Gerald Jay Sussman, my thesis supervisor, for his unending support throughout these years, for being a constant source of inspiration, and for his encouragement for pursuing scholarly perfection. Gerry has taught me that work and fun can be combined into one.

Harold Abelson, my thesis supervisor, for teaching me mathematical thinking and for his excellent critique of my writing. Hal has helped me debug and clarify many of the ideas of this thesis.

George Verghese, my thesis reader, for providing good advice and critical feedback throughout my thesis project.

David Waltz, my thesis reader, for helpful discussions and his advice on my professional career.

Paul Penfield, my academic counselor, for his encouragement and advice on my graduate studies.

Rich Zippel (now at Cornell) and John Wyatt, for unfaltering faith in my work since the first year of my graduate study at MIT; Richard Thornton, for introducing me to the maglev application; Weiping Li, for teaching me essentials of nonlinear control; John Guckenheimer, for teaching me dynamical systems theory while I was visiting the Center for Applied Mathematics at Cornell in the summer of 1990; Bob Hermann, Marc Raibert, Elisha Sacks, and Jean-Jacques Slotine for helpful discussions.

Franklyn Turbak, for being a wonderful friend and a source of fun; Mark Sheldon and Pierre Jouvelot for providing kindly distraction from my work; Andy Berlin, Mike Eisenberg, Chris Hanson, Bill Rozas, Liz Bradley, Becky Bisbee, and other members of MIT Project on Mathematics and Computation for making the Lab a fun place to work.

Ken Yip and Arthur Gleckler for being wonderful officemates over the years and for stimulating discussions.

And most of all, Ying Yin, for her love and encouragement through all these years. This thesis would never have been completed without the moral support and love from my family.

This thesis describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-89-J-3202, and in part by the National Science Foundation grant MIP-9001651. The author was also supported by a Chu Fellowship through the MIT Graduate School.

Contents

1 Introduction **1**
1.1 Contributions 2
1.2 Goal: Mechanizing Control Design 4
1.3 Limitations of Traditional Methods 5
1.4 Characteristics of the Approach of This Thesis 6
1.5 What the Control Engineer's Workbench Does 7
1.6 Organization of the Thesis 10

2 The Control Engineer's Workbench **11**
2.1 Reasoning About Control Design 11
2.2 A Real Scenario with the Workbench 14
2.3 Overview of the Workbench 17
  2.3.1 Design requirements 17
  2.3.2 Anatomy of the Workbench 19

3 Automatic Phase-Space Modeling and Analysis — MAPS **24**
3.1 Introduction 24
3.2 Qualitative Phase-Space Structures 26
  3.2.1 Equilibria, limit cycles, and stability regions 26
  3.2.2 Trajectory flows 28
3.3 Automated Qualitative Analysis of Phase-Space Structures 29
  3.3.1 Theoretical characterization of stability regions 30
  3.3.2 Extracting and representing shapes of stability regions . . . 31
  3.3.3 Modeling trajectory flow pipes 41
3.4 The MAPS Analysis Algorithm 47
  3.4.1 The algorithm 47
  3.4.2 The main illustration 50

3.4.3 Implementation details 58
3.4.4 More examples 58
3.4.5 Hierarchical extraction and representation of phase-space information 61
3.5 Discussion 63
3.5.1 Scope of the analysis 63
3.5.2 Extensions 65
3.5.3 The use of MAPS in visualization 66
3.6 Summary of the Chapter 66

4 Automatic Phase-Space Control Synthesis — Phase Space Navigator 68
4.1 Introduction 68
4.2 Automatic Control Synthesis in Phase Space 69
4.2.1 Overview of the Phase Space Navigator 69
4.2.2 Intelligent navigation in phase space 69
4.2.3 Planning control paths with flow pipes 70
4.3 The Phase Space Navigator 72
4.3.1 Reference trajectory generation 72
4.3.2 Reference trajectory tracking 74
4.3.3 The autonomous control synthesis algorithms 76
4.3.4 Discussion of the synthesis algorithms 78
4.4 An Example: Stabilizing a Buckling Column 81
4.4.1 The column model 82
4.4.2 Extracting and representing qualitative phase-space structure of the buckling column 84
4.4.3 Synthesizing control laws for stabilizing the column 85
4.4.4 The phase-space modeling makes the global navigation possible 92
4.4.5 Other control problems 95
4.5 Summary of the Chapter 95

5 An Application: Design of a Maglev Controller 97
5.1 Introduction 97
5.2 The Maglev Model 98
5.3 State-Space Control Trajectory Design 100
5.3.1 Modeling state-space geometry 100

5.3.2 Synthesizing a global stabilization law 101
5.3.3 Evaluating the control design 106
5.3.4 Visualizing the design 108
5.3.5 Implementation of the controller 108
5.4 Summary of the Chapter 112

6 Related Work 113
6.1 Qualitative Analysis of Dynamics 113
6.2 Engineering Stability Analysis 115
6.3 Phase-Space Nonlinear Control 115
6.4 Intelligent Control 117

7 Conclusions 119
7.1 Thesis Revisited 119
7.2 Problems and Future Work 120
7.3 Broad Implications 122
7.4 Towards A Language for Computational Control Design 123
7.5 The Big Picture 124

A Maglev Application: Equilibria 126

B Maglev Application: Synthesized Reference Trajectories 127

List of Figures

1.1 The high-level description of the control design for a maglev system, automatically generated by the Workbench: (a) a projection of the controllable region showing several synthesized reference trajectories and the operating equilibrium; (b) a graphical rendering of the controllable region; (c) a symbolic summary characterizing the system and the control design. 9

2.1 The phase plane of the buckling column (from Abraham & Shaw 1988). 13
2.2 Engineer Input: the model for a buckling steel column. 14
2.3 Workbench Output: a symbolic summary of the analysis. 15
2.4 Workbench Output: the phase portrait of the buckling column showing equilibrium points, boundaries of stability regions, and a flow pipe leading to the left-hand attractor. 16
2.5 Engineer Input: the control objective. 16
2.6 Workbench Output: the control reference trajectory and the control law for stabilizing the column. In the plot, the reference trajectory is drawn in solid lines, and the flow-pipe boundaries of the uncontrolled column in dashed lines. 18
2.7 The structure of the Workbench. 21
2.8 The flow of computation in the Workbench. 23

3.1 Equilibrium points: (a) attractor, (b) repellor, and (c) saddle. . . . 27
3.2 A stability region (basin of attraction) for an attractor in a 2nd-order system. 28
3.3 A trajectory flow. 29
3.4 Stability boundaries for an attractor: (a) non-degenerate boundary — separatrix; (b) degenerate boundary. 32

3.5 The stability boundary for an attracting limit cycle. 33
3.6 A Delaunay triangulation over a set of points in a plane. 36
3.7 Circumcircle properties of exterior triangles vs. interior ones: (a) a region with smooth shape; (b) a region with narrow-parts. 39
3.8 Grouping trajectories into flow pipes: (a) a collection of trajectories with the same qualitative feature; (b) a corresponding flow pipe. . . 43
3.9 Classifying faces of simplices with respect to flows. 44
3.10 Construction of flow pipes: (a) grouping simplices to form a monotonic polyhedron by canceling common non-monotonic faces; (b) aggregating monotonic polyhedra to form a flow pipe according to flow directions at the boundaries. 45
3.11 Topological shapes of flow pipes. 47
3.12 The flow chart of MAPS. 49
3.13 The main illustration: the vector field of a 2nd-order nonlinear system. 51
3.14 The main illustration—MAPS output: (a) equilibrium points; (b) boundary and connecting trajectories. (to be continued) 53
3.15 (cont'd) The main illustration—MAPS output: (c) points on the stability boundary for one of the attractors; (d) triangulation of the convex hull. 54
3.16 (cont'd) The main illustration—MAPS output: (e) polyhedral approximation to the stability region computed from the triangulation on boundary points; (f) two flow pipes computed from the refined triangulation. The flow pipes form the stability region. 55
3.17 The main illustration: the relational graph constructed by MAPS. . 56
3.18 The main illustration—MAPS output: a symbolic summary. 57
3.19 The analysis of a 3rd-order nonlinear system: (a) projection of stability boundary and connecting trajectories in x-z plane; (b) projection of polyhedral approximation in x-z plane. 60
3.20 A 2nd-order system with a saddle connection. 62
3.21 A multi-layered representation for a dynamical system used in MAPS. 63

4.1 Search for a control path from an initial point to a goal point in a stack of phase portraits. 71
4.2 The Phase Space Navigator 72
4.3 Reference trajectory generation 73
4.4 Reference trajectory tracking 75

4.5 The flow chart of the trajectory planning algorithm of the Phase Space Navigator. 79
4.6 Buckling of a thin elastic steel column due to axial end loads. . . . 82
4.7 The buckling column: the phase space of a buckling column showing the stability boundaries and connecting trajectories. The horizontal axis x is the characteristic measure of the column displacement from its principal axis and the vertical axis x' is the velocity. 83
4.8 The buckling column: the flow pipe leading to the attractor on the left. 85
4.9 The flow-pipe graph for the buckling column. 87
4.10 The goal projection and the deformation of the trajectory. 88
4.11 The synthesized control law that stabilizes the buckling column: (a) the reference trajectory that leads to the unbuckled state corresponding to the saddle at the origin. The column is initially buckling with sufficient velocity; (b) the position x of the column plotted against time t. (to be continued) 90
4.12 (cont'd) The synthesized control law that stabilizes the buckling column: the velocity $v(=x')$ of the column and the control signal u for stabilizing the column are plotted against time t in (c) and (d), respectively. 91
4.13 The synthesized control law for restoring the column from the buckled state: (a) the reference trajectory that swings the column out of the buckled state; (b) the position x of the column plotted against time t. (to be continued) 93
4.14 (cont'd) The synthesized control law for restoring the column from the buckled state: the velocity $v(=x')$ of the column and the control signal u for controlling the column are plotted against time t in (b) and (c), respectively. 94
4.15 The mechanical model for an overhead crane unloading cargo from ships, reproduced from [Sakawa&Shindo, 1982]. The crane is equipped with a trolley drive motor and a hoist motor. The planar motion of the load is modeled as a swinging pendulum hanging at the moving trolley. 96

5.1 EMS maglev system for high-speed ground transportation, representing a simplified drawing of the German Transrapid design. (a) Electromagnetic suspension (from [MTAC Report 1989]); (b) Detail of a suspension magnet, superimposed on the field distribution (from [Eastham 1989]). 99
5.2 Stable and unstable manifolds of the saddle for $V_i = 140$ (yz-projection). 102
5.3 Stable and unstable manifolds of the saddle for $V_i = 300$ (yz-projection). 103
5.4 The sandwiched region in yz-projection. 104
5.5 The boundaries of the sandwiched region in yz-projection: (a) top boundary; (b) side boundary; (c) bottom boundary. 105
5.6 The synthesized control reference trajectories originating from four different initial states, together with the controllable region: (a) yz-projection; (b) zx-projection. 107
5.7 Simulation of the nonlinear control design for different initial displacements. (to be continued) 109
5.8 (cont'd) Simulation of the nonlinear control design for different initial displacements. 110
5.9 The controllable region for the maglev control design, rendered with the light source straight on. Hidden faces and lines are removed. . . 111

List of Tables

3.1 The internal representation of phase-space data objects. 59

Chapter 1

Introduction

> The purpose of computing is insight, not numbers.
>
> — R. W. Hamming

Synthesis and analysis of control systems for complex physical systems are difficult tasks. Complex systems operate in large nonlinear regimes; they often comprise many components and contain a large number of state variables. Linear systems have been well studied, and there are systematic design procedures for linear control systems. On the other hand, nonlinear systems are far less understood. Nonlinear control synthesis is limited by the lack of available control schemes and analysis techniques; traditional design and analysis are seldom applicable to these systems.

Powerful computers can help engineers and scientists unveil the intricacies of nonlinear phenomena and can revolutionize the design of complex, high-performance control systems. The computational exploration of nonlinear systems has drastically changed the way we think about the world. This opens the possibility of simulating and designing physical systems, without resorting to linear approximations.

Yet despite advances in computer technology, nonlinear control systems are still difficult to design and analyze. The difficulties arise from the lack of design methodologies that actively exploit the special nature of nonlinearities, and from the lack of high-level computational tools that can harness the available computational power. Few nonlinear systems admit closed-form solutions. The ability to understand and control these systems requires extensive numerical simulations. The burden of the task is on human engineers to carefully design experiments, to distill qualitative information from numerical simulation results, and to use the information to design appropriate control actions. This manual, numerical approach is burdensome and error-prone, and severely limits the design space one can explore.

The modern geometric theory of dynamical systems, pioneered by Poincaré at the beginning of the century, provides a qualitative way to describe the rich dynamical behaviors of nonlinear systems. The geometric representation forms a viable substrate for computationally analyzing qualitative aspects of dynamics and exploring novel control-synthesis strategies. Programs that employ powerful symbolic and numerical techniques have already shown promise in the geometric analysis of nonlinear dynamical systems [2].

However, the phase-space geometry and topology of a dynamical system are hard to describe in a way that allows for efficient computational manipulations for the purpose of control design. Nonlinear systems can have extremely convoluted phase-space geometries; the complexity becomes much worse as the dimensionality increases. Humans can comfortably picture and manipulate two and three-dimensional objects with the aid of graphic, geometric modeling techniques. For higher-dimensional systems, however, few visualization and manipulation tools exist. Automatic modeling, analysis, and design tools are necessary to identify, extract, and reason about the spatial properties of phase space.

1.1 Contributions

This thesis demonstrates that difficult control synthesis tasks can be automated, using a computational workbench consisting of programs that actively exploit knowledge of nonlinear dynamics and phase space. These programs combine powerful numerical and symbolic computations with spatial reasoning techniques. The thesis contributes to the state of the art in artificial intelligence and control systems engineering in several ways:

- The thesis has developed a **qualitative representation** for complex behaviors of dynamical systems in phase space and a **design language** for computationally expressing and manipulating these behaviors. The qualitative representation captures the gross aspects of dynamics in a relational graph of phase-space structure and a set of discrete objects called *flow pipes*—the equivalence classes of behaviors. The design language describes a control design task in terms of well-defined geometric, combinatorial operations on the flow pipes. This language helps formalize aspects of implicit expert reasoning of control engineers in solving control design problems. The representation and the language are developed independently of the orders of systems, *i.e.*, the dimensionality of phase spaces.

- The thesis has constructed a computational environment, **the Control Engineer's Workbench**, integrating a suite of programs that automatically analyze and design high-performance, global controllers for a large class of nonlinear systems. These programs combine powerful techniques from numerical and symbolic computations with novel representation and reasoning mechanisms of artificial intelligence. The two major components in the Workbench—the Phase Space Navigator and MAPS[1]—work together to visualize and model the phase-space geometry and topology of a given system. They reason about and manipulate the phase-space geometry and topology and search for optimal control paths connecting initial state and the desired state for the system. The Workbench represents the result of design and analysis in a symbolic form manipulable by other programs, and produces a high-level summary meaningful to professional engineers. It also presents the result in a graphical form.

- The thesis presents a novel phase-space **design methodology** for the synthesis of automatic controllers. The design methodology computationally exploits dynamical systems' nonlinearities in terms of phase-space geometries and topology. It designs a prespecified control law—control reference trajectories—for a system by synthesizing the desired shape for phase-space geometry dictating trajectory flows. It uses the novel technique of flow pipes to group infinite numbers of distinct behaviors into a manageable discrete set that becomes the basis for establishing reference trajectories, and navigates the system along the planned reference trajectories. The phase-space design approach requires powerful computational tools that are able to identify, extract, represent, and manipulate qualitative features of phase space, and is embodied in programs comprised in the Workbench.

---

[1] MAPS stands for Modeler and Analyzer for Phase Spaces.

- The thesis has demonstrated the Workbench and the phase-space design methodology in an **application** of great practical interest: the Workbench helped design a high-quality controller for a magnetic levitation system—the German Transrapid system. The control system synthesized by the Workbench stabilizes a maglev vehicle with much larger initial displacements than those allowed in a previous linear design for the same system using classical linear feedback technique. The simulation shows that our design outperforms the linear design by a factor of 20. Professional control engineers consider this result important enough to present at the 31st IEEE Conference on Decision and Control [56].

1.2 Goal: Mechanizing Control Design

A real-world control system is a complex closed-loop system with extremely rich dynamics. Computation and reasoning are pervasive in the design and operation of the controller. Sensors collect a large amount of quantitative information. State and parameter estimators infer hidden information about the system from the sensed data. The system is modeled with a representation appropriate for further analysis and design based on available information. The model is then analyzed to extract behaviors that are judged significant for the control objective. To meet the control objective, a control law is synthesized to change the natural dynamics of the system.

The domain of automatic control brings together issues of sensing, estimation, control synthesis, and control execution. The study of their common themes—**computation** and **reasoning**—serves as a framework for coherently addressing these issues and makes it possible to employ advanced computational techniques to drastically improve modern control design. This thesis focuses on the control synthesis.

A control engineer goes through the following design steps to synthesize a controller for a given physical system:

1. **Modeling:** translate the physics and the constraints of the physical system into a quantitative mathematical model.

2. **Analysis:** analyze the model of the system.

3. **Design:** design a controller for the system.

4. **Verification:** verify the control design. Iterate Steps 2–4 if necessary.
5. **Implementation:** implement the control design in software or hardware.

In the above control design procedure, Step 1 models the system, usually with a set of differential equations. Step 2 examines the model and analyzes the behaviors of the system. Based on the analysis, Step 3 arrives at a control design according to the prespecified control constraints. Step 4 ensures that the design meets the control specification. The last step implements the control design to control the real physical system. The Steps 2, 3, and 4 are often iterated before a reasonable control law is synthesized. Except for very few cases in which analytic-form solutions are available, computer simulations are the main tools for analyzing the systems and for designing and verifying the controllers.

> *The task of control design and analysis requires powerful knowledge representation and reasoning mechanisms and computer simulation techniques. The mechanization of this task and the exploration of novel control design methods are the subjects of this thesis research.*

1.3 Limitations of Traditional Methods

We are interested in developing a computational design method suitable for dealing with the complexities of real-world nonlinear control systems and in mechanizing the design process by computer programs. Traditional methods relying on purely analytic or numeric techniques or on linear design theory are inadequate for carrying out and automating the design process.

Systematic techniques for control design of linear systems have been well developed. Controllers for linear systems can be synthesized via methods such as pole-placement or Bode diagrams [35], with satisfactory performance. However, complex control systems operate in larger regions where system nonlinearities are too significant to ignore. These nonlinear control systems do not lend themselves easily to linear (small signal) analysis and design. There have been attempts to extend linear control techniques to nonlinear control systems: a piecewise-linear control design, also called gain scheduling, approximates a nonlinear function with several linear pieces, each of which admits a linear control design. For example, in aircraft autopilot design, an airplane system model is linearized around several hundred operating points selected within the plane's operating region. Each operating point is subject to a linear controller design. However, this type of control can be very expensive and complex in both design and implementation. For a highly nonlinear system operating in a large region, an enormous number of linearizations and controller designs have to be carried out. The complexity and the cost of conventional implementations prohibit many practical applications.

The computational method developed in this thesis designs a nonlinear control system in phase space. Unlike gain scheduling, our method explicitly models and actively exploits the nonlinearities in terms of trajectory flows and can carry out control design automatically. Although phase space is an extremely useful medium for exploring novel nonlinear design methods, the phase-space design approach requires sophisticated computational techniques and representational mechanisms to make the approach computationally feasible.

Existing control simulation softwares are inadequate for automatically designing highly complex nonlinear systems. Commercially available programs like MATLAB and SIMULAB [30] rely on numerical simulations. These programs are, at their very best, semi-automatic and serve as interactive design aids to human engineers. Although they are equipped with elaborate graphic interfaces, these programs provide only fragmented, limited capabilities such as integrations and equilibrium determination for performing the simulation task; human users need to prepare the simulation and to interpret the result. The specialized control toolboxes embedded in these programs are "shallow" expert systems; they lack deep domain knowledge and do not have mechanisms for computationally representing and manipulating a control design.

1.4 Characteristics of the Approach of This Thesis

The computational approach presented in this thesis has the following characteristics: it can analyze and design nonlinear, global control laws; it is autonomous; and its development is independent of the order of a system.

Our approach exploits the richness of nonlinear dynamics and phase-space representations. For example, the programs use the flow pipes to explore a variety of control paths otherwise difficult to obtain with just a piecewise linearization. Our approach designs control trajectories from a global point of view and thereby maximizes the effect of control actions to obtain good performance.

The programs embody significant knowledge of dynamical systems theory and control theory. Because of their autonomous nature, the programs can perform certain tasks that human counterparts have difficulty with, for instance, the task of visualizing and manipulating complicated high-dimensional geometric objects. The machinery for geometric modeling is developed independently of the spatial dimensionality. Hence the programs are not restricted to low-order systems, as long as the computational complexity allows.

These programs are mostly useful in synthesizing high-performance control systems that do not lend themselves to traditional design and analysis techniques. On the other hand, the qualitative analysis could also assist engineers in exploring novel control design methods. Aside from the utility of these programs, this thesis formulates the task of control design in a computational way; this formulation provides a concrete substrate and a rich domain for exploring and developing new reasoning and representation techniques otherwise hard to grasp in traditional computer science and artificial intelligence research.

1.5 What the Control Engineer's Workbench Does

Given a model of a physical system and a control objective, the Control Engineer's Workbench analyzes the system and designs a control law achieving the control objective. A user typically interacts with the Workbench in the following way.

The user first tells the Workbench about the system: he inputs a system model in terms of a differential equation, parameter values, and bounds on state variables for analysis in the form of a phase-space region. The user also tells the Workbench about the requirements on the control design: he specifies the desired state for the system to settle in, the initial states of the system, the allowable control parameter values, and the constraints on the control responses.

The user then asks the Workbench to analyze the system within the parameter ranges of the model. The Workbench visualizes the totality of the behaviors of the system over the parameter ranges; it represents the qualitative aspects of the system in a data structure and reports to the user a high-level, symbolic summary of the system behaviors and, if necessary, a graphic visualization of the phase-space qualitative features.

Next, the user instructs the Workbench to synthesize a control law for the system, subject to the specified design requirements. The Workbench searches for the global control paths that connect the initial states of the system and the desired state, using the qualitative description about the system. More specifically, the search is conducted in a collection of discrete entities representing trajectory flows in phase space. After the global control paths are established, the Workbench determines the controllable region of the system and the switching surfaces where control parameters should change values. A synthesized control reference trajectory consists of a sequence of trajectory segments, each of which is under a constant control. For a point-to-point control design, the Workbench further constructs a smoothed trajectory connecting an initial state and the desired state.

The Workbench has designed a high-performance nonlinear control law for an electro-mechanical system of practical interest—the stabilization of the German Transrapid magnetic levitation transportation system. The details of this design will be described in Chapter 5. Here, we briefly illustrate what the Workbench does in this application.

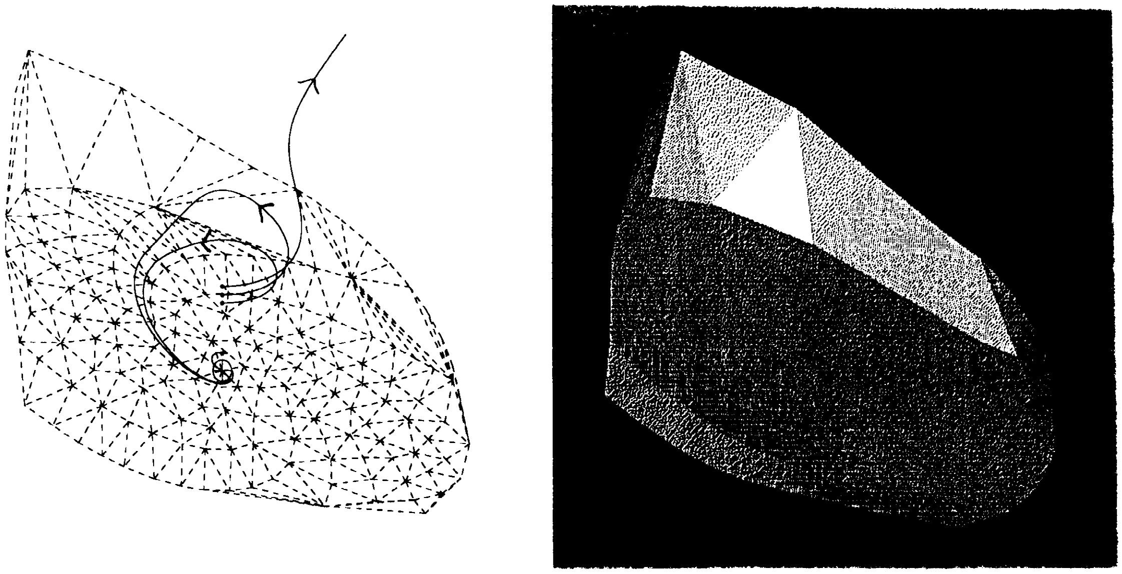
The train and guideway system of the Transrapid is described by a third-order nonlinear differential equation:

$$\begin{cases} \frac{dx}{dt} = \frac{z(V_i - Rx)}{L_0 z_0} + \frac{xy}{z} \\ \frac{dy}{dt} = g - \frac{L_0 z_0 x^2}{2mz^2} \\ \frac{dz}{dt} = y \end{cases}$$

Ensuring smooth ride-quality for passengers is one of the top priorities in the design of the Transrapid system. Unfortunately, the system is unstable at the operating equilibrium for parameter V_i at 140 volts. The objective of the control design is to synthesize a control law that stabilizes the system at the operating equilibrium.

However, the stabilization control of the train, typically traveling at about 350 miles per hour along the guideway subject to various disturbances, is a very difficult task, for the system operates in a large operating region where nonlinearities come from the geometries of the train and the guideway and from the magnetic forces. The system requires a high-quality global stabilization design.

The Workbench is given the system equation in a symbolic form, the ranges of parameter values, a bounding box in phase space, and the control objective. It analyzes the behaviors of the system for the given parameter values and synthesizes a controllable region about the previously unstable operating equilibrium; the region is maximum with respect to the allowable control authority.



(a) (b)

```
<equilibrium-points:
 equilibrium 1. (saddle at (140. 0. 200.))>

<flow-pipes:
 flow-pipe 1. (from *infinity* to *infinity*)
   boundary: (2D-manifold)
 flow-pipe 2. (from *infinity* to *infinity*)
   boundary: (2D-manifold)>

<controllable-region:
 interior: 831 tetrahedra
 boundary: 526 triangles
 volume: 0.848
 range-of-control: (140 300)
 number-of-switching-surfaces: 1
 maximum-displacement-z: 0.00455>
```

(c)

Figure 1.1: The high-level description of the control design for a maglev system, automatically generated by the Workbench: (a) a projection of the controllable region showing several synthesized reference trajectories and the operating equilibrium; (b) a graphical rendering of the controllable region; (c) a symbolic summary characterizing the system and the control design.

The Workbench reports the results of the analysis and the control design in a graphic and symbolic description in Figure 1.1. The description contains graphic displays of the controllable region showing the region boundary, the equilibrium, and synthesized control reference trajectories from sampled initial states; it also characterizes the behaviors of the system and the controllable region in a symbolic form. Although the pictures shown here are in black-and-white, the actual screen-display for the region produced by the Workbench is in color. The result of this application has been presented to professional control engineers and been favorably compared with a previous manual, linear design for the system [56].

1.6 Organization of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 first gives an overview of the Control Engineer's Workbench. The bulk of the thesis is then divided into three parts: Chapter 3 develops an analysis method, MAPS, for constructing qualitative representations of dynamics in terms of asymptotic behaviors and equivalence classes, *i.e.*, the flow pipes; Chapter 4 presents Phase Space Navigator for the synthesis of control laws in phase space. The Phase Space Navigator searches for optimal control trajectories in a collection of flow pipes forming a flow-pipe graph. Chapter 5 applies the machineries developed in Chapters 3 and 4 to an engineering problem of practical interest: the design of a global, nonlinear controller for the stabilization of a magnetic levitation transportation system. Finally, Chapter 6 describes other related work in the computational analysis and design for control systems, and Chapter 7 concludes the thesis with a summary and future work in the area.

Although the domain this thesis studies is control design, readers with knowledge of linear algebra, elementary differential equations, and an introductory to linear systems or control will be able to go through the thesis with little difficulty. I have included necessary background materials on dynamical systems theory and control theory to make the thesis self-contained.

Chapter 2

The Control Engineer's Workbench

This chapter gives an overview of the Control Engineer's Workbench—an implemented computational environment for analyzing and designing dynamical control systems in phase space. The Workbench, consisting of a suite of programs, allows engineers to visualize the phase-space structures of the systems, to manipulate the dynamics, and to synthesize controllers.

2.1 Reasoning About Control Design

How does a control expert reason about a control design task? A professional control engineer uses a body of specialized knowledge to carry out the design. The engineer develops insight through an analysis of the physical system, uses the insight to explore the design space constrained by control requirements, and makes engineering judgment about design choices and trade-offs. A particularly intuitive design method is the phase-plane method for analyzing and designing a 2nd-order nonlinear control system in a phase plane [35].

Let us consider an example of the deformation of an elastic column. Putting a weight on top of the column, the column oscillates around its principal axis. Putting a heavier weight, the column oscillates and settles to a buckled state after a short while. How would an engineer analyze the behavior of the system and design a control law to stabilize the buckling motion with the phase-plane method?

The engineer first describes the system with a mathematical model. Compiling the dynamics into a suitable model is a nontrivial task. Discussion of model building is outside the scope of this thesis. We assume that the engineer is given a simple model for the column in the form of an equation of motion, a 2nd-order nonlinear ordinary differential equation. Starting with the model and a set of typical parameter values, the engineer determines all possible responses of the system within certain ranges of amplitudes and input signals. The responses with different initial conditions from the simulation are then plotted in a two-dimensional plane—the phase plane (Figure 2.1). A state, a pair of position and velocity, is plotted as a point in the plane. A trajectory consists of a sequence of consecutive states and is drawn as a one-dimensional curve. With this graphical plot, the engineer identifies the natures and locations of critical points. He then inspects the gross features of trajectories: the trajectories approach two critical points in spirals, respectively. The engineer groups the trajectories into two classes, each of which corresponds to those approaching the same critical point. The phase plane gives a qualitative picture of the system responses. The shaded region in the picture is the collection of trajectories going to the left-hand critical point, one of the buckled states. The critical point at the middle of the picture is the unbent state of the column. In order for the engineer to interpret the responses, he needs to know things like system states, trajectories, critical points, and qualitative regions.

The buckling motion that leads the column to a buckled state traces out a trajectory in the phase plane, for instance, a trajectory that starts at a location in the shaded region and evolves to the left-hand critical point in Figure 2.1. Suppose the engineer wants the column to return to its original unbent state after the oscillation dies down, instead of letting it get buckled. He introduces a control input, a force exerting on the column, and observes how the system responses vary with the input. With the pictorial description of the responses, the engineer determines a solution curve in phase space as a control law that moves the trajectory to the middle critical point.

In short, the engineer goes through the following steps with the phase-plane method:

- simulate the system extensively with different initial conditions
- plot the behaviors in trajectories on a piece of paper
- interpret the result with visual inspection

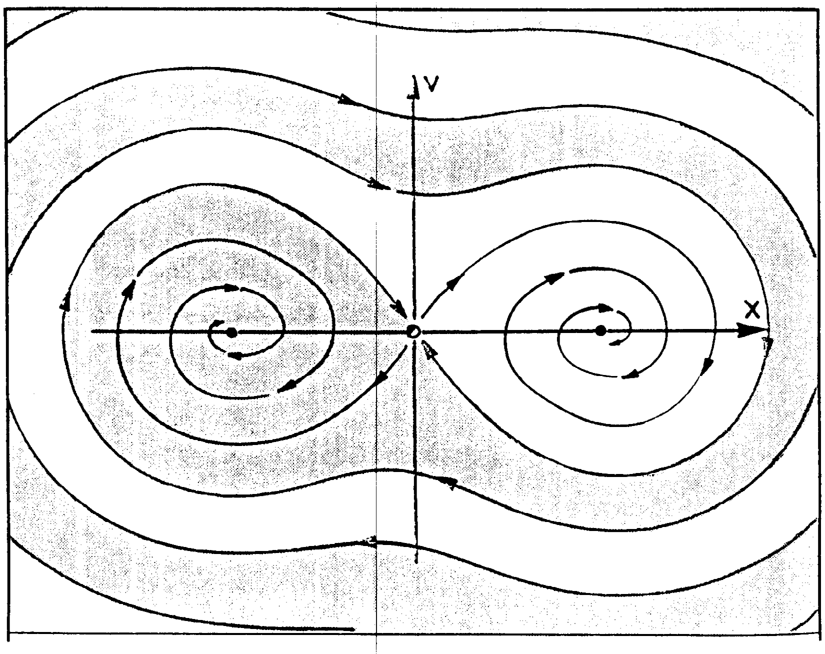


Figure 2.1: The phase plane of the buckling column (from Abraham & Shaw 1988).

- design a control law to obtain desired behavior.

The phase-plane method is able to analyze the transient and asymptotic behaviors of planar systems, whether linear or nonlinear. Kalman had used this method to design switching control laws based on the phase-plane plot [26]. His method decomposes a phase plane into discrete linear regions, designs linear control law for each region, and matches transients on the boundaries of the regions.

The phase-plane method is a useful tool for analyzing qualitative responses of a control system. It is, however, manual, prohibitively expensive, and confined to planes. Although the diagrammatical sketch of a phase plane illustrates just the qualitative aspects of the system, to obtain a picture like this requires lots of human effort in preparing numerical simulations, collecting the numbers, sketching the results, analyzing the trajectory plot, and interpreting it in a qualitative picture like Figure 2.1. Worse, this method becomes useless in cases when the order of a system is greater than two and the nonlinearity results in convoluted phase-space geometry. The mechanization of the task with autonomous computer programs would alleviate many of these restrictions.

`name:`	`buckling_column`
`equation_of_motion:`	$\begin{cases} dx_1/dt = x_2 \\ dx_2/dt = -p_1 x_1 - p_2 x_1^3 - p_3 x_2 + u \end{cases}$
`state_variable:`	x_1
`state_variable:`	x_2
`parameter:`	$p_1 =$ `-2.0`
`parameter:`	$p_2 =$ `1.0`
`parameter:`	$p_3 =$ `0.2`
`parameter:`	$u =$ `0.0`
`bounding_box:`	$x_1 \in$ `[-3.0,3.0]`, $x_2 \in$ `[-4.0,4.0]`

Figure 2.2: Engineer Input: the model for a buckling steel column.

2.2 A Real Scenario with the Workbench

The Control Engineer's Workbench automates a significant portion of the control engineer's design task. In the following session, the Workbench autonomously analyzes the buckling motion of a steel column under compression and synthesizes a control law to stabilize the motion. The interaction between a human user and the Workbench is annotated in explanatory italics for ease of reading.

- *The engineer types in the model of the elastic column buckling under axial compressive force and asks the Workbench to analyze the system for the given parameter values (see Figure 2.2).*
- *The Workbench summarizes the qualitative behaviors of the system in terms of equilibria, stability regions, and flow pipes (see Figure 2.3).*

  The Workbench also displays the phase portrait of the system showing the first flow pipe (see Figure 2.4).
- *The engineer instructs the Workbench to design a control law that stabilizes the buckling motion, in particular, a law that brings the column back to the unbuckled state corresponding to the unstable equilibrium at the center of Figure 2.4. He specifies the control design requirements (Figure 2.5).*

```
<equilibrium-points:
 equilibrium 1. (attractor at (1.41 0.))
 equilibrium 2. (saddle at (0. 0.))
 equilibrium 3. (attractor at (-1.41 0.))>

<trajectories:
 <boundary-trajectories:
  trajectory 1. (from *infinity* to (0. 0.))
  trajectory 2. (from *infinity* to (0. 0.))>
 <connecting-trajectories:
  trajectory 3. (from (0. 0.) to (-1.41 0.))
  trajectory 4. (from (0. 0.) to (1.41 0.))>>

<stability-regions:
 stability-region 1.
   attractor at *infinity*
   stability-boundary: ()
   connecting-trajectories: ()
 stability-region 2.
   attractor at (1.41 0.)
   stability-boundary: (trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 4)
 stability-region 3.
   attractor at (-1.41 0.)
   stability-boundary: (trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 3)>

<flow-pipes:
 flow-pipe 1. (from *infinity* to (-1.41 0.))
   boundary: (trajectory 2. trajectory 1. trajectory 3.)
 flow-pipe 2. (from *infinity* to (1.41 0.))
   boundary: (trajectory 2. trajectory 1. trajectory 4.)>
```

Figure 2.3: Workbench Output: a symbolic summary of the analysis.

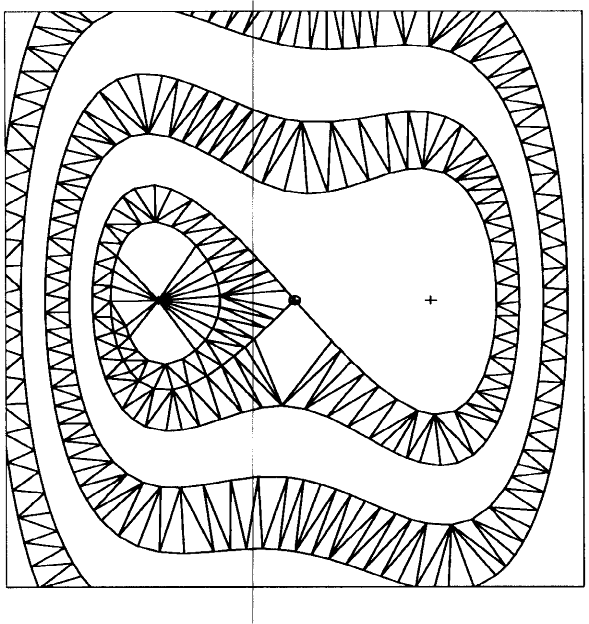


Figure 2.4: Workbench Output: the phase portrait of the buckling column showing equilibrium points, boundaries of stability regions, and a flow pipe leading to the left-hand attractor.

control_type:	point_to_point
goal_state:	(0.0, 0.0)
initial_state:	(-1.0, -3.0)
range_of_control:	$u \in$ [-0.2,0.2]

Figure 2.5: Engineer Input: the control objective.

- *The Workbench reports a synthesized control reference trajectory (see Figure 2.6 for a phase-space illustration and a script of the control law). The global portion of the reference trajectory shown in the figure consists of four segments, each of which starts at a switching state marked as a small circle. The control parameter value is held constant for each segment and is denoted by* $U1$, $U2$, $U3$, *or* $U4$. *The control law for the reference trajectory is represented as a sequence of tuples: (time, switching-state, control). The control law invokes a local controller when the trajectory enters the vicinity of the goal. The Workbench determines that the control response time for the global portion of the reference trajectory is less than 7 seconds.*

2.3 Overview of the Workbench

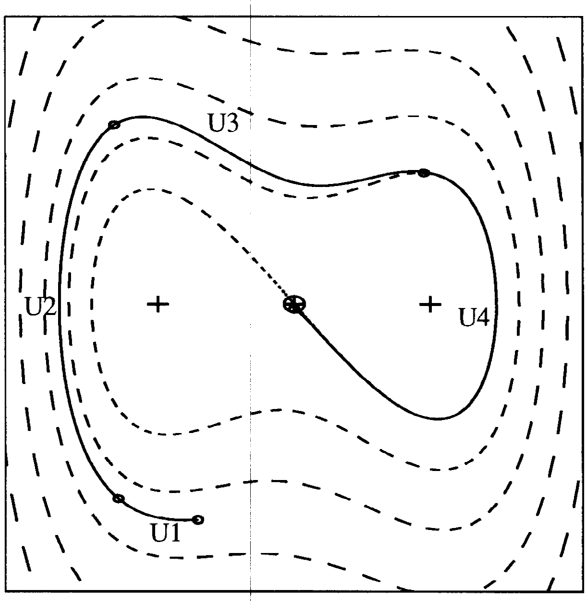
In the previous session with the Workbench, the Workbench has demonstrated the following capabilities:

- Deciding what behaviors are significant. The Workbench looks for qualitative features like equilibrium points, stability regions, and trajectory flows.
- Describing the behaviors qualitatively in computational terms. The Workbench models the stability regions and trajectory flows geometrically.
- Reasoning about a phase space with flow pipes.
- Performing the control design autonomously.

These capabilities are supported in the Workbench by (1) qualitative representation consisting of dimension-independent constructs, (2) hierarchical extraction of the behaviors, (3) modeling and manipulation mechanisms for flow pipes, and (4) algorithms implementing the geometric, combinatorial, and numerical computations.

2.3.1 Design requirements

We motivate the design criteria for the Workbench with the intended task domain of control synthesis and with computational considerations.



```
;; The Synthesized Control Law: specifying the time instance, switching
;; state, and corresponding control value for each switching:

((time 0.) (switching-state #(-1 -3)) (control .2))
((time .284) (switching-state #(-1.82 -2.71)) (control 0.))
((time 1.06) (switching-state #(-1.86 2.49)) (control -.2))
((time 2.71) (switching-state #(1.35 1.82)) (control 0.))
((time 6.76) (switching-state #(-.0023 -.0692)) (control *local-control*))
```

Figure 2.6: Workbench Output: the control reference trajectory and the control law for stabilizing the column. In the plot, the reference trajectory is drawn in solid lines, and the flow-pipe boundaries of the uncontrolled column in dashed lines.

- The design of the Workbench calls for a concise representation that captures essential features of a control system. The representation should be sensible to human engineers and manipulable by other programs. The Workbench needs to present the result of its analysis and design to human designers and to encourage the designers to interact with the design in a direct way. This communication requires a high-level, intuitive presentation of the result. Other programs in the Workbench need to efficiently access and manipulate the representation. Instead of encapsulating everything about the system, the representation should contain only information that is useful for the control synthesis task. We have chosen a qualitative representation describing a system in terms of its phase-space geometric features for this purpose.

- The Workbench needs modeling algorithms to efficiently construct the representation. The algorithms should identify and extract implicit dynamical properties from numerical explorations and summarize the result in the qualitative form. In the implementation, the Workbench internally uses a hierarchy of intermediate representations. The qualitative information about the system is extracted in a step-by-step fashion, from local descriptions to global ones.

- The Workbench also needs a reasoning mechanism to efficiently manipulate the representation for synthesizing a control law. The program searches through the representation to find feasible control trajectories. The Workbench uses a graph mechanism that manipulates a discrete collection of objects called the flow pipes.

2.3.2 Anatomy of the Workbench

The Workbench analyzes and designs control systems in phase space. In the scenario in Section 2.2, the Workbench locates equilibria and models stability regions; it groups trajectories, *i.e.*, behaviors, into the flow pipes; it constructs a dimension-independent representation for the phase space; it designs control reference trajectories with this representation. In addition, the Workbench communicates with the user in high-level terms through an interface for inputting the model and for describing the design and analysis and a graphic module for displaying the result.

Like the phase-plane analysis, the Workbench is able to reason about and manipulate dynamics in terms of phase-space geometries. It partitions a phase space into discrete regions. For the previous elastic column example, the program autonomously explores the behaviors of the system and generates the high-level picture of the phase portrait in Figure 2.4. The picture contains essentially the same kind of information as that of Figure 2.1.

But unlike the phase-plane method, the Workbench provides computational means for manipulating the dynamics. It decomposes the phase space into subregions that can even be globally nonlinear. The shaded region in Figure 2.4 is approximated as a geometric polygon. The program internally represents the critical points and geometries of regions in a data structure that allows other programs to manipulate, visualize, and communicate with human users. Because the geometry and topology of a system's phase space are modeled with a simplicial structure, the representation and reasoning mechanisms for this structure are independent of the dimensionality of phase space.

The Workbench serves as an intelligent assistant to control engineers. The components of the Workbench are shown in Figure 2.7:

- the MAPS program for simulation and interpretation
- the Phase Space Navigator for control synthesis
- a graphic program for visualizing the design
- a user interface for communication with the system.

The component for model building is not in the Workbench yet. The programs in the Workbench implement various algorithms: symbolic differentiation, numerical algorithms on differential equations, numerical integrations, modeling of geometric structures, clustering of equivalence classes, graph algorithms, etc. The inference mechanism of the Workbench uses these programs to construct a qualitative phase-space structure for representing a system, to check for the consistency of the structure, and to reason about and manipulate the representation through a graph of flow pipes.

The flow of computation within the Workbench is illustrated in Figure 2.8. Given a model of a system, a bounded phase-space region of interest, allowable parameter values, and control objectives and constraints, the Workbench performs stability and trajectory flow analysis for the system in phase space and interprets the result in a phase-space graph. The Workbench then explores the control space to synthesize a desired control law subject to the design constraints. It reports

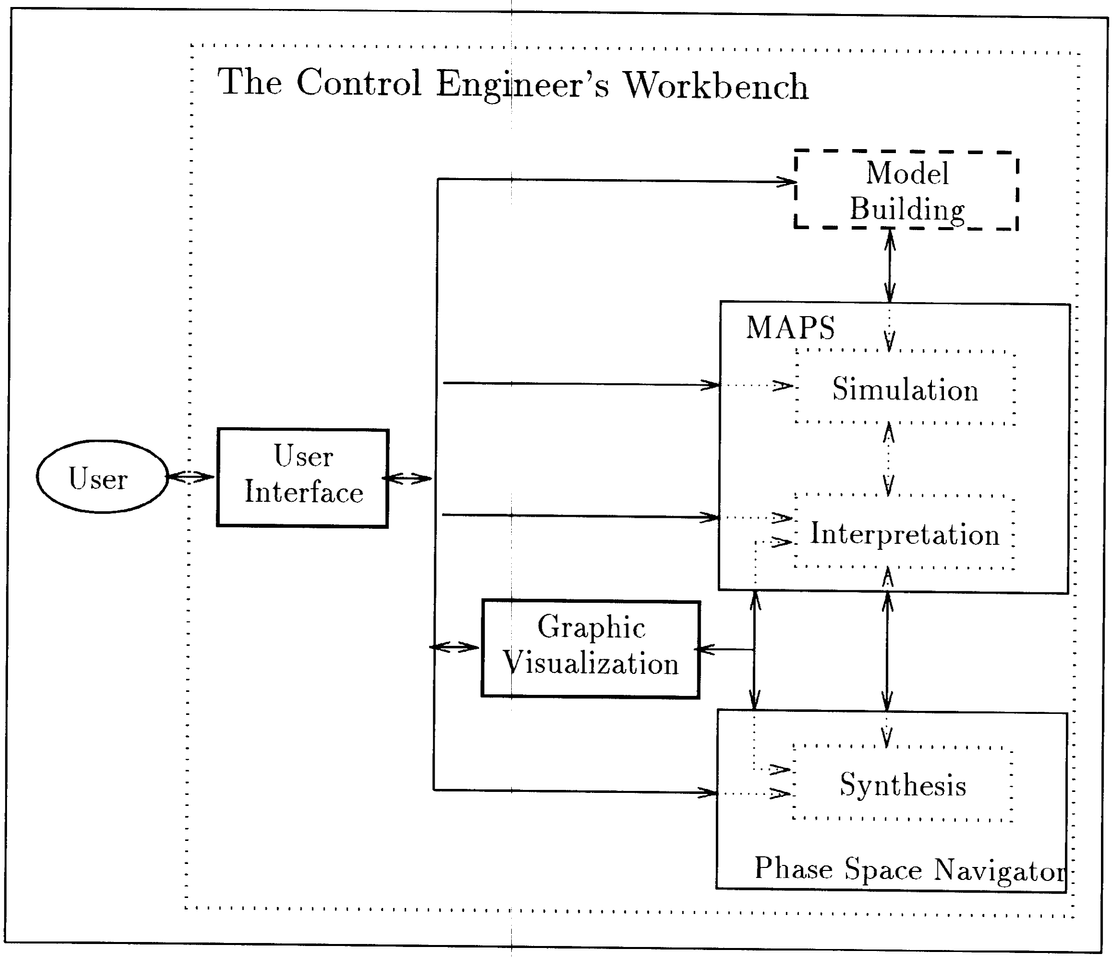

The Control Engineer's Workbench
Model Building
MAPS
Simulation
Interpretation
User
User Interface
Graphic Visualization
Synthesis
Phase Space Navigator

Figure 2.7: The structure of the Workbench.

the control law specifying reference trajectories and performance properties. The control design for steering towards an equilibrium is performed in this way: for a point-to-point control, the output is a reference trajectory, whose control law consists of a sequence of tuples of time, switching state and the corresponding control value; if an initial operating region is given, the output is a controllable region geometrically represented as a polyhedral structure. The steering towards a limit cycle has not been implemented yet, although its implementation would be very much the same in Poincaré sections as the case for steering towards an equilibrium.

The programs in the Workbench are written in Scheme, a dialect of LISP. The Workbench is implemented on HP 700 series workstations. For moderately difficult problems, for example, the buckling steel column example, it takes the Workbench on the order of minutes to arrive at a control law.

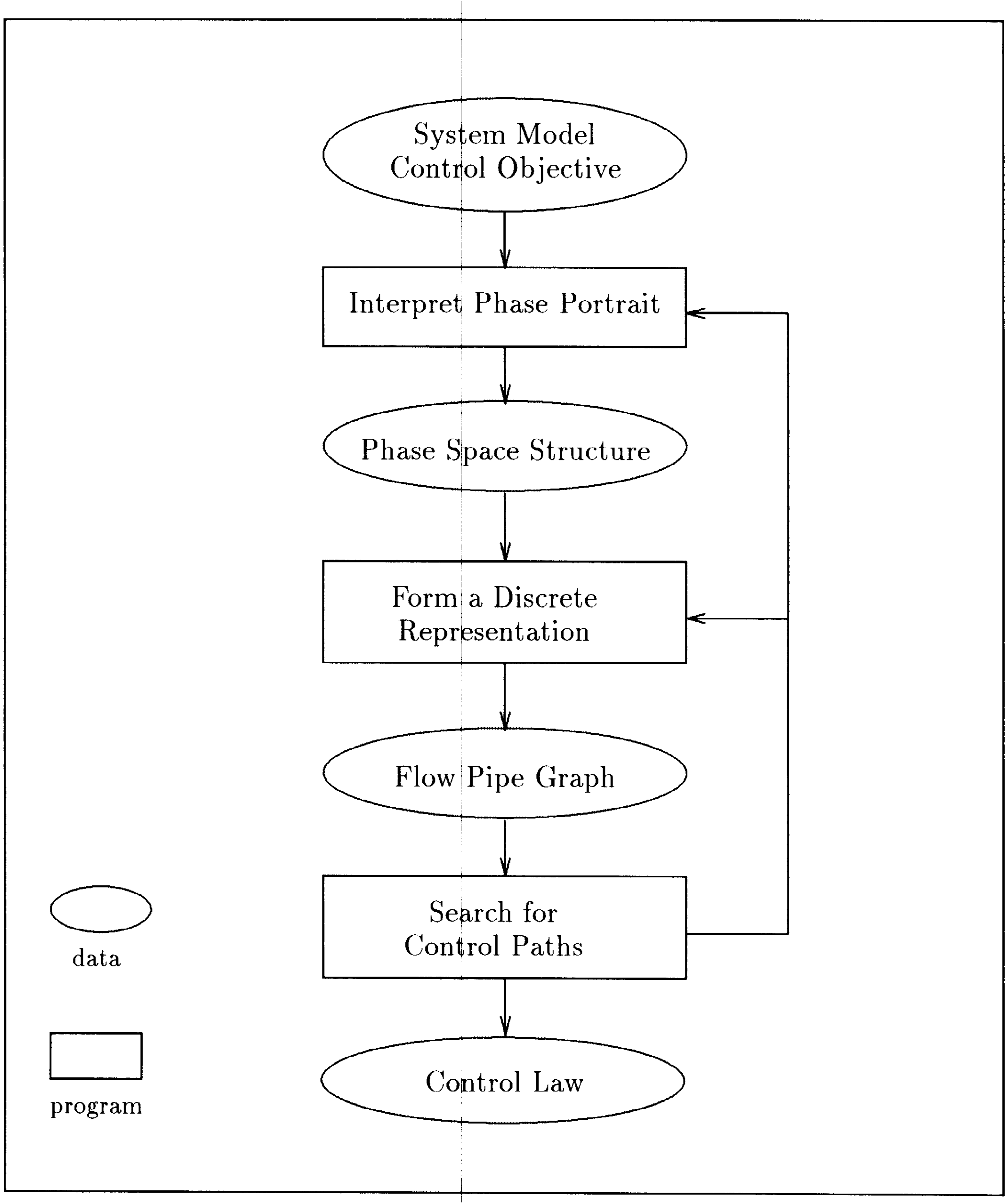

System Model
Control Objective
Interpret Phase Portrait
Phase Space Structure
Form a Discrete Representation
Flow Pipe Graph
Search for Control Paths
Control Law
data
program

Figure 2.8: The flow of computation in the Workbench.

Chapter 3

Automatic Phase-Space Modeling and Analysis — MAPS

This chapter develops and demonstrates a computational method for automatically analyzing qualitative behaviors of dynamical systems in phase space. We will discuss the extraction and representation of the qualitative behaviors of a dynamical system in terms of a qualitative phase-space structure describing spatial arrangement of geometric entities in phase space, a geometric construct called *flow pipe* for modeling trajectory flows, an analysis algorithm and its implementation in the MAPS program, and examples illustrating the mechanism of the analysis.

3.1 Introduction

Analysis of dynamical systems via phase-space geometric structures plays an increasingly important role in experimenting, interpreting, and controlling complex systems [1, 2]. Nonlinear systems usually fall outside the domain of traditional analysis methods, such as Fourier analysis for linear systems. However, most of the important qualitative behaviors of a nonlinear system can be made explicit in phase space with a geometric analysis [39].

MAPS is a program for understanding and representing qualitative structures of phase spaces. MAPS combines numerical, symbolic, and geometric computations with techniques of spatial reasoning. It embodies theoretical knowledge about nonlinear dynamical systems and formulates in computational terms a significant amount of informal working knowledge of professional control engineers in analyzing complex control systems, in particular, in the form of the phase-plane method [35]. We will illustrate our techniques of extracting and representing the qualitative phase-space features with two- and three-dimensional systems. The techniques presented in this chapter also apply to higher-dimensional dynamical systems.

Complex systems are often nonlinear and high dimensional. Our theoretical knowledge about nonlinear dynamical systems is far from complete. Therefore, many engineering applications rely on extensive numerical experiments. A numerical simulation typically generates an immense amount of quantitative information about a complex system. To interpret the numerical result and to use the information for engineering designs, it is essential to develop qualitative methods that automatically analyze the system, extract the qualitative features, and represent them in a high-level description sensible to human beings and manipulable by other programs. The representation for the qualitative behaviors needs to be parsimonious and yet capture the essential properties of the dynamical system under study. The task of control design discussed in the next chapter requires that the representation facilitate efficient manipulation in synthesizing new dynamical behaviors. We have chosen an equivalence-class based qualitative representation motivated by the above requirements on the form and use of the representation.

Our goal is to develop a class of intelligent and autonomous programs that understand the behaviors of complex systems through their phase-space representations, synthesize control commands, and affect the physical processes. For example, a controller that controls the locomotion of an autonomous walking robot would monitor the state of the system, analyze the motion, and command the motor to achieve a particular walking behavior. When these systems operate in nonlinear regimes and are of high order, their complexities often defy human analysis. In order for the robot to autonomously execute control tasks with superior performance, it is important that the robot understands the consequences of its control action and synthesizes appropriate control commands to affect the state of the system. We are particularly interested in automating the control analysis and synthesis for a class of nonlinear systems that do not lend themselves easily to traditional analysis and design techniques.

3.2 Qualitative Phase-Space Structures

We are interested in representing the qualitative behaviors of dynamical systems for control analysis and design. One useful qualitative representation of the phase space of a dynamical system is in terms of equilibrium points and limit cycles, stability regions, trajectory flows exhibiting the same qualitative features, and the spatial arrangement of these geometric objects. We call this representation the qualitative phase-space structure for a dynamical system and give the following definition.

Definition 3.1 Qualitative phase-space structure: *The qualitative phase-space structure within a phase-space region of interest, for a structurally stable dynamical system addressed in this thesis*[1]*, consists of*

(a) the number, positions, and stability types of equilibrium points and limit cycles

(b) the geometric structures of stability regions (basins of attraction) associated with the attractors

(c) the equivalence classes of trajectory flows

(d) the spatial arrangement of the equilibrium points, limit cycles, stability regions, and trajectory flows.

3.2.1 Equilibria, limit cycles, and stability regions

We review some of the basic concepts in dynamical systems theory in order to describe the qualitative phase-space structures. Let us consider a single pendulum perturbed from its downward resting position. It swings around its vertical axis with a smaller and smaller amplitude, and eventually settles to the resting position due to friction. We call such a resting state a stable *equilibrium point.* We say that an initial state of the pendulum is in the *stability region* of the stable equilibrium point if the pendulum starting from that state eventually settles to the stable equilibrium point. In this example, all initial states that the pendulum starts from except for the vertically upward one are in the stability region.

---

[1] The scope of the class of systems is discussed in Section 3.5.1.

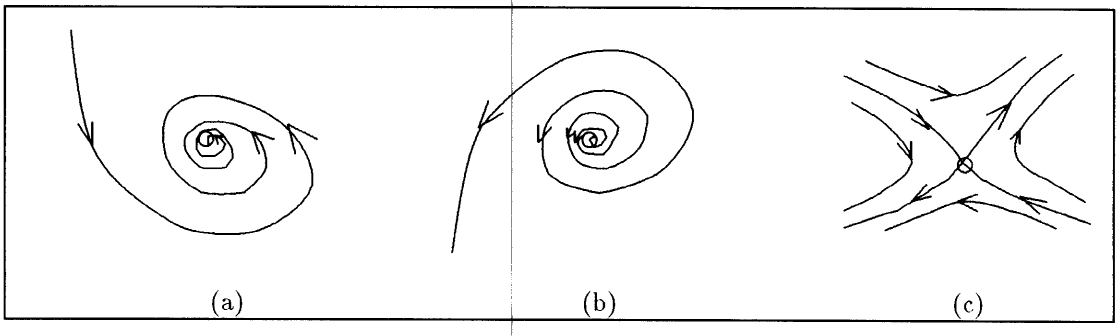


Figure 3.1: Equilibrium points: (a) attractor, (b) repellor, and (c) saddle.

In general, the equilibrium points of a dynamical system $x' = f(x, u)$, where u is a parameter, are the zeros of the vector field $f(x, u) : R^n \rightarrow R^n$. Structurally stable systems [18] can have equilibrium points of three types: *attractor*, *repellor*, and *saddle*, whose local behaviors in phase spaces are shown in Figure 3.1. An attractor is an equilibrium point that all nearby trajectories approach in forward time. In the pendulum example, the downward resting state is an attractor. We call the attractor an asymptotically stable equilibrium point. A repellor is the one that repels all nearby trajectories. One can think of it as an attractor in reverse time. Trajectories approach a saddle in some directions and leave it in the other directions. The vertically upward state of the pendulum is a saddle. Start the pendulum resting at this position. A slight perturbation would move it from the position. On the other hand, the pendulum with just the right amount of energy could swing to the upward position and rest there, although it will be extremely rare and will take an infinite amount of time to do so. We call the repellors and saddles unstable equilibrium points. The study of equilibria plays a central role in dynamical systems theory. The equilibrium points, together with other phase-space entities, partition a phase space into qualitatively different subregions. The attractors are the only kind of equilibria that can be observed physically; they characterize the asymptotic behaviors of dynamical systems. The other classes of steady-state behaviors addressed in this thesis are limit cycles.

The important concept of stability is associated with the stability regions. The collection of trajectories approaching an equilibrium point is called the *stable trajectories* (or *stable manifold*) of the point; and the collection of trajectories leaving an equilibrium point is called the *unstable trajectories* (or *unstable manifold*) of the

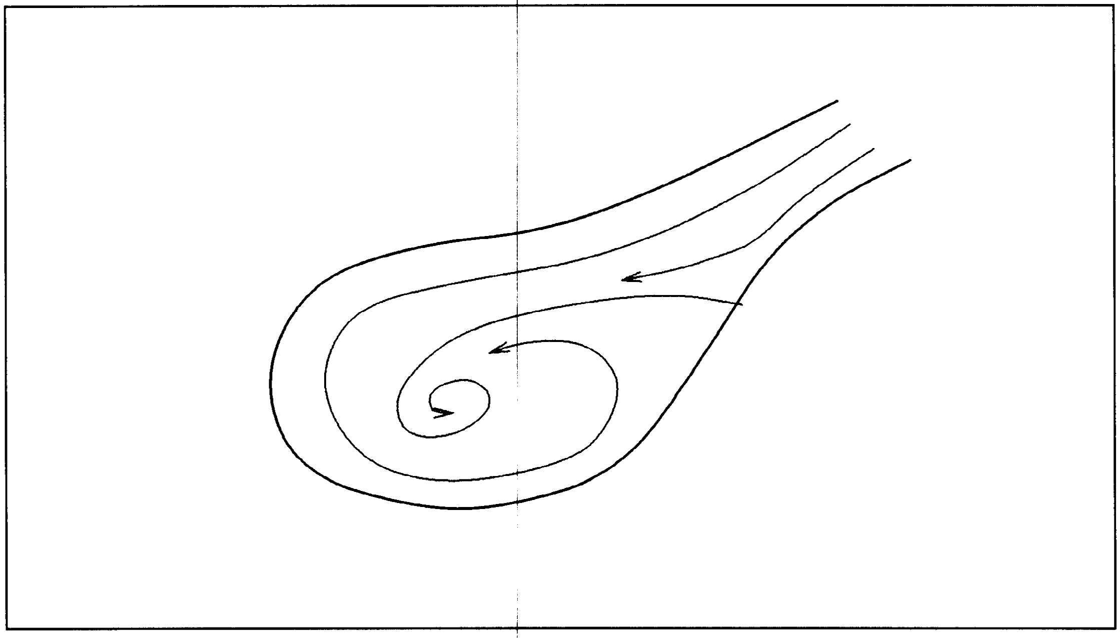


Figure 3.2: A stability region (basin of attraction) for an attractor in a 2nd-order system.

point. A saddle has stable trajectories along some directions and unstable trajectories along the other directions. The union of the stable trajectories of an attractor is its stability region, often called the basin of attraction for the attractor. The stability region of an attractor is open. Every trajectory starting in the stability region will be attracted to the attractor, by definition. Attractors cannot be on the boundary of the region. The region may be either bounded or unbounded; it is unbounded if the boundary contains no repellors. An example of a stability region is shown in Figure 3.2.

3.2.2 Trajectory flows

Most nonlinear systems do not have solutions that can be expressed in closed form. Analysis of such systems often resorts to computational exploration of their trajectories over a period of time. Since a phase space is a union of an infinite number of trajectories, an exhaustive search for plausible control paths in the space of all the trajectories is computationally too expensive and numerically too sensitive to uncertainties. To make it possible to find good control paths, we bundle

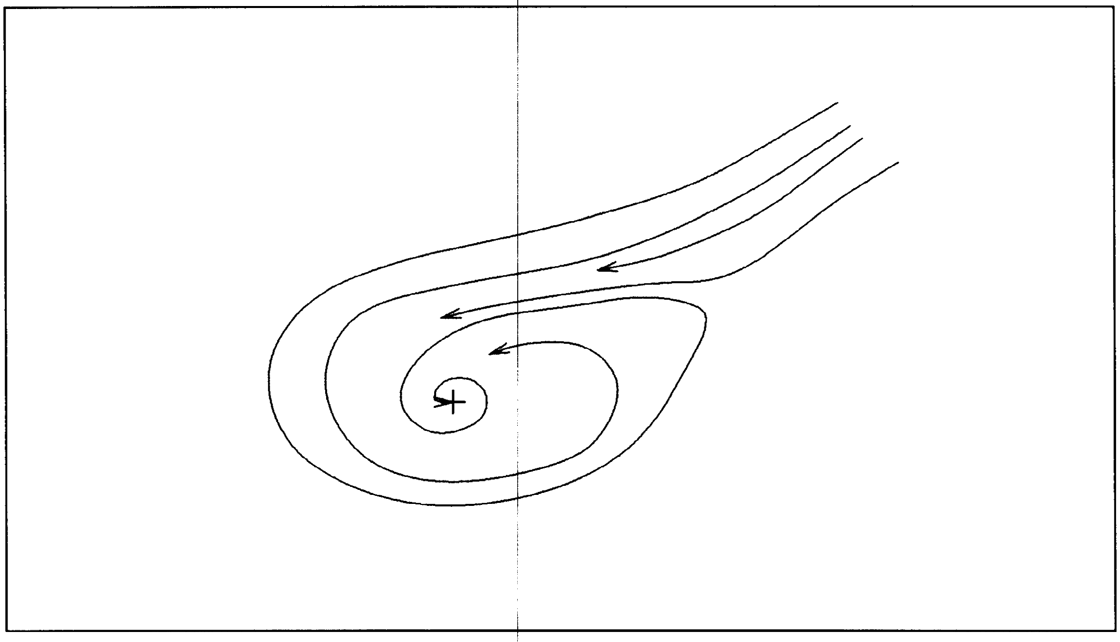


Figure 3.3: A trajectory flow.

together trajectories with similar qualitative features into a trajectory flow.

The trajectory flows describe the collective shape and direction of the trajectories. Figure 3.3 shows, for example, a trajectory flow in the phase space consisting of all trajectories ending at the same destination marked by the symbol +.

With the trajectory flow bundle representation of the phase space, the exploration is constrained in a much smaller space of equivalence classes of trajectories. The expensive fine-grain search in phase space, possibly of high dimensions, is avoided. Furthermore, the flow bundle is a natural representation with respect to the robustness to the effect of noise and uncertainties, compared with individual trajectories. The effect of noise on an individual trajectory is modeled by *thickening* the trajectory.

3.3 Automated Qualitative Analysis of Phase-Space Structures

MAPS understands qualitatively different regions and extracts and represents geometric shape information about these regions. Given a dynamical system specified as a system of governing equations, MAPS generates a high-level symbolic description of the phase-space structure as the result of the analysis. The high-level description will be used as input to other programs for further computations, for example, for synthesizing control laws in the next chapter.

3.3.1 Theoretical characterization of stability regions

An essential part of the stability analysis for a dynamical system is the determination of the boundaries of stability regions—the stability boundaries. Our algorithm for determining stability boundaries is based on a crucial theoretical result characterizing the stability boundaries of a fairly large class of dynamical systems[2]. Under certain weak conditions to be discussed in Section 3.5.1, the result of Chiang et al. [10] shows that the stability boundary for an attractor consists of the stable trajectories (stable manifold) of equilibrium points and limit cycles whose unstable trajectories (unstable manifold) approach the attractor. This allows us to numerically determine a collection of trajectory points on the stability boundary through calculations of the stable and unstable trajectories[3]. In planar systems, the boundaries consist of curve segments obtained from trajectories. In higher dimensions, the boundary surfaces are swept out by a set of trajectories on the boundary. The following theorem establishes the theoretical basis for the stability analysis of the algorithm.

Theorem 3.1 Characterization of stability boundaries for attractors *[Chiang et al. [10]]: For nonlinear autonomous dynamical systems satisfying certain conditions, the stability boundary for an attractor is the union of stable manifolds of equilibrium points and limit cycles, whose unstable manifolds approach the attractor.*

The three conditions that the systems have to satisfy will be discussed later in the thesis. The equilibria that can be on the stability boundaries are saddles and repellors. A boundary that separates stability regions for different attracting sets is called a separatrix. A degenerate boundary is one that does not separate stability regions.

---

[2] Our algorithm does not use Liapunov functions to construct stability regions. The class of Liapunov function based methods will be discussed in Section 6.2.

[3] The trajectory points can be obtained, for example, from numerical integration algorithms.

Figure 3.4(a) shows the stability boundary for the attractor labeled by the rightmost +. One of the unstable trajectories of the saddle, denoted by $\oplus$, approaches the attractor as $t \to \infty$. Therefore, the stable trajectories labeled "b" of the saddle form the stability boundary for the attractor. Figure 3.4(b) shows a stability region with degenerate boundary labeled "b".

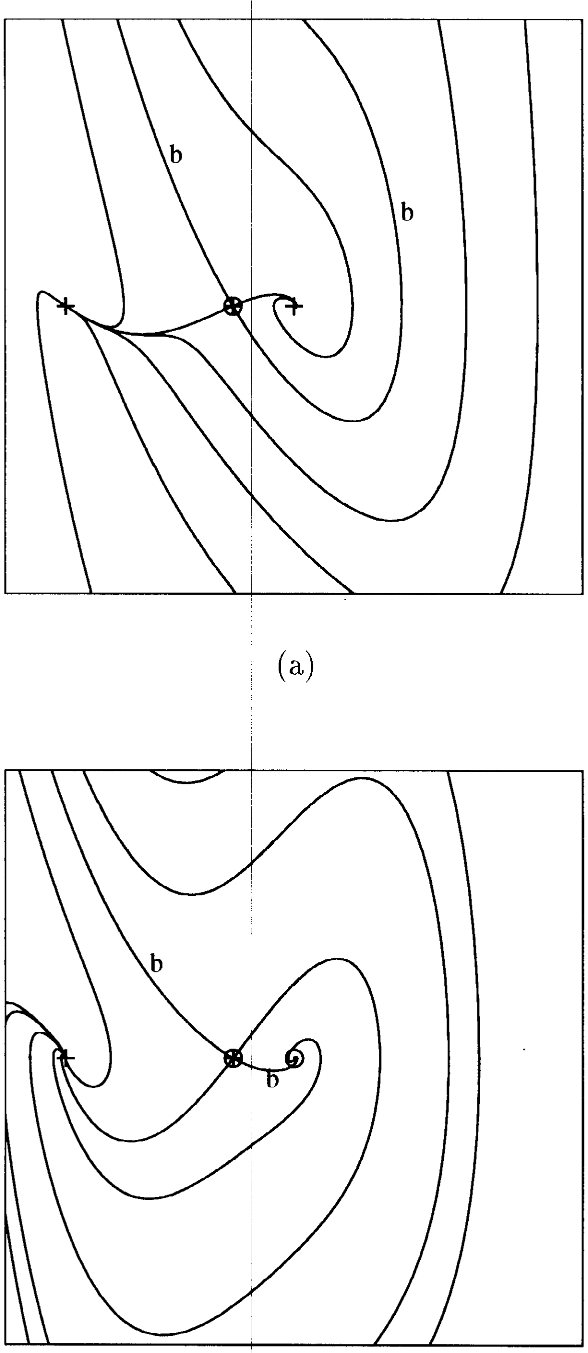
Parker and Chua have generalized Theorem 3.1 to characterize stability boundaries for attracting limit cycles [38]. The corresponding result remains much the same as Theorem 3.1 except that the occurrences of "attractor" in Theorem 3.1 are substituted with the words "attracting limit cycle". Figure 3.5 schematically illustrates the stability boundary for an attracting limit cycle. The boundary is formed by two repelling limit cycles. Parker and Chua's generalization is known to be true for many important examples. However, it is largely an intuitive generalization; no rigorous proof has been given yet. Although our techniques for extracting geometries of stability regions for point attractors are readily applicable to those of attracting limit cycles, we shall not discuss them in detail here.

3.3.2 Extracting and representing shapes of stability regions

Since a phase space of a nonlinear dynamical system often consists of qualitatively distinct points and regions, the phase-space "shape" of the system refers to the geometric information about the structures and spatial arrangements of these points and regions. A qualitative analysis of the dynamical system determines the "shape" of the phase space. The geometric information about these regions, for example, is extremely useful in analyzing stability properties of control designs for complex dynamical systems, such as electric power systems and mechanical control systems.

Definition 3.2 The "shape" of a dynamical system: *The "shape" of a dynamical system refers to the geometric characterization of the qualitatively distinct points and regions of the phase-space representation of the system.*

We need to extract the geometric information about the stability regions from the numerical results about the stability boundaries, and represent it parsimoniously so as to facilitate further computations. For example, the representation is to be used to estimate the volumes of the stability regions, to reason about the



(a)

(b)

Figure 3.4: Stability boundaries for an attractor: (a) non-degenerate boundary — separatrix; (b) degenerate boundary.

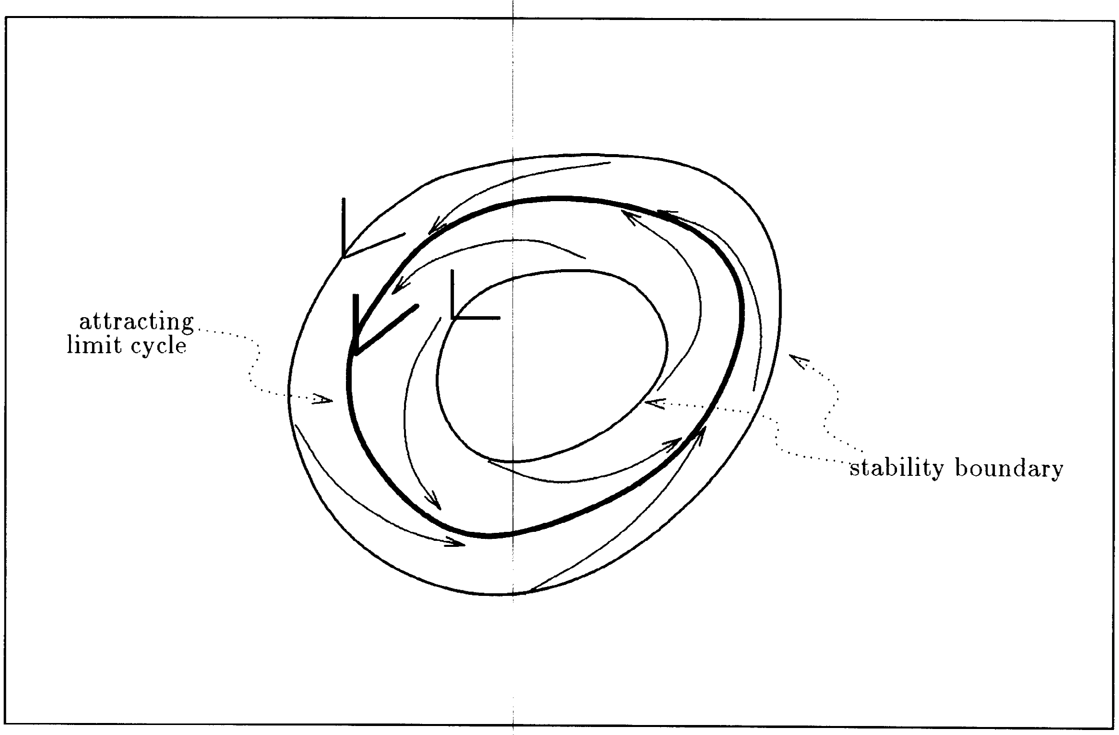

attracting limit cycle
stability boundary

Figure 3.5: The stability boundary for an attracting limit cycle.

spatial relations with the stability boundaries, to compute topological properties of the regions, to extract geometric information about trajectory flows, etc.

Generation of discrete trajectory points

The boundary of a stability region is numerically approximated by a collection of trajectory points. A one-dimensional boundary curve is approximated by a sequence of points on boundary trajectories, a two-dimensional boundary surface is approximated by a collection of points on boundary trajectories that sweep out the surface, and so on.

The discrete trajectory points are generated with numerical integration method. The method of integrating trajectories from initial points is well-know. However, it is difficult to generate a set of trajectory points that evenly populate a surface. The difficulty arises from deciding where the initial points come from and when the integration stops.

Consider for example a two-dimensional stability-boundary surface. Since the stability boundary is formed by stable manifolds of saddles, the trajectories sweeping out the surface can be generated by integrating the trajectories backwards from initial points near the saddles in the directions of stable eigenvectors. This integration method also generalizes to higher-dimensional boundary surfaces.

In order to maintain relatively even spacings between surface trajectories, the program must select a subset from all the possible initial points and terminate an integration of a trajectory when the trajectory gets too close to other trajectories. We use the following criteria for this purpose: (1) the distribution of trajectory points, (2) inter-trajectory distances, and (3) local curvature of the surface.

Let's take the surface of a sphere as a stability boundary, and assume a saddle is at the north pole and a source at the south pole. As we integrate backwards from the saddle along a set of surface trajectories, these backward trajectories diverge as they get near the equator. If the distance between two adjacent trajectories is too large, an additional trajectory is needed to cover the gap. The program backtracks in this case: it picks an initial point near the north pole, *i.e.*, the saddle, to generate this new trajectory. The local curvature of the surface also helps determine if an additional trajectory is necessary. As the backward trajectories begin to converge at the south pole, some of the trajectories can be terminated for the purpose of approximation.

A simplicial representation

Given a set of trajectory points on the stability boundary, a structure on the points is needed to make explicit the proximity relationships among these points on the boundary. We define the structure on the trajectory points as a graph: the vertices are the given trajectory points; the edges join those points that are related to each other, where the notion of relation is defined by some metric. Among the different structures on the points, a minimal representation—one with fewest edges and preserving topological structure—is a polyhedral structure having those boundary points as vertices. Furthermore, the resulting polyhedron is contained in the convex hull of the boundary points. This polyhedron is not unique. We choose the one computed from the Delaunay triangulation to be discussed shortly.

A polyhedral structure is a consequence of the so-called simplicial representation. The elements of a simplicial representation consist of simplices [33]. An n-dimensional simplex, or an n-simplex for short, determined by $n+1$ geometrically independent points[4] is the convex hull of the points. These points act as vertices of the n-simplex. A familiar example of a 0-simplex is a point, a 1-simplex is a closed line segment, a 2-simplex is a closed triangle, a 3-simplex is a solid tetrahedron, and so on.

The simplices are basic building blocks from which we construct complicated geometric spaces. There are certain ways simplices can be joined to form a new object; for instance, two triangles can be glued together at a vertex, or along an edge to form a quadrilateral. Given a set of points in a space, a simplicial tessellation, or a triangulation, is a collection of simplices that cover the convex hull of the given points and have these points as vertices. For example, a triangulation on a set of points on a two-dimensional surface divides up the surface into triangular regions.

Among possible triangulations over these points, the Delaunay triangulation [40] offers certain advantages over the others with respect to the quality of triangles measured by the regularity of the triangular mesh. The Delaunay triangulation has the property that the circumsphere of any simplex does not contain any vertices in its interior. This property makes the Delaunay triangulation very attractive: the

---

[4] A set of points $\{x_0, x_1, ..., x_n\}$ of R^N ($N \geq n$) is said to be geometrically independent, if and only if the vectors $x_1 - x_0$, ..., $x_n - x_0$ are linearly independent. Thus two distinct points in R^1 form a geometrically independent set; so do three non-collinear points in R^2 and four non-coplanar points in R^3.

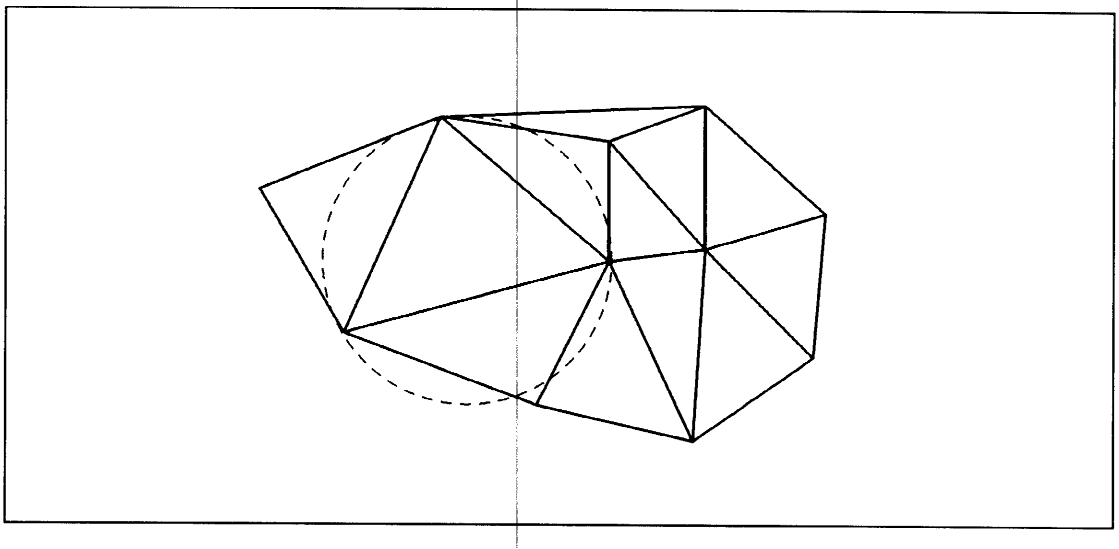


Figure 3.6: A Delaunay triangulation over a set of points in a plane.

simplices in the triangulation are most equiangular over all triangulations; in two dimensions, the minimum interior angles of its triangles is maximized. Figure 3.6 shows the Delaunay triangulation on a set of points.

We are concerned with deriving a discrete representation for the geometric structure of a phase space that computer programs can efficiently reason about. The representation needs to preserve the gross features of shape of the structure with minimum complexity. Yip has used a point-set representation for the phase-space geometries and topologies in analyzing two-dimensional Hamiltonian maps [52]. While the point-set representation has been successfully used to recognize patterns of the phase portraits, it is not suitable for the purpose of cutting and pasting trajectory flows and synthesizing desired phase-space geometry and topology. The simplicial structure, on the other hand, is a much coarser representation for describing qualitative features of geometric pieces. The simplicial representation is independent of the dimensions, permits recursive decomposition, and is characterized by certain algebraic signatures such as fundamental groups [33]. A point-set representation or trajectory representation would be too fine-grain for manipulating the shape of phase spaces and would incur unnecessary complexity in searching for control paths in phase space.

Extraction of geometric structures

The geometric information about a stability region is represented by the polyhedron tightly stretched over the trajectory points on the stability boundary. Extraction of this polyhedral approximation proceeds in two steps: computing a triangulation of the convex hull containing the polyhedron, and eliminating exterior triangles. The triangulation is computed with the Delaunay method on the set of points which results in a tessellation of the convex hull of these points with simplices. The polyhedral approximation is then extracted from the triangulation by a sculpture method used in visual information representation [7], followed by a more expensive centroid method. Simplices exterior to the polyhedron are eliminated first by heuristic rules and then by testing their membership in the stability region with trajectories emanating from the centroids of the simplices.

We have implemented the Delaunay triangulation in n dimensions that supports incremental insertion and deletion of points, adapted from an algorithm proposed for planar triangulation [37]. The time complexity of the algorithm grows exponentially with the dimensions, so does the number of simplices in the triangulation. For N points in n spatial dimensions, the time and space complexities scale as $N^{O(n)}$ [40].

Under the condition that the distribution of boundary points is reasonably dense and uniform on the boundary, the polyhedron approximating the stability region is contained in the triangulation of the convex hull. Furthermore, the union of all the circumspheres of interior simplices is a good approximation for the region. In order to extract the polyhedron, simplices exterior to the polyhedron have to be eliminated.

For geometric structures with relatively smooth surfaces, *i.e.*, small curvatures, the exterior simplices are eliminated with a relatively fast and inexpensive method using the circumsphere heuristics [7]. We observe that only certain type of simplices that satisfy the so-called "visibility conditions" can be exterior ones and are therefore candidates for elimination. We associate a value V with each candidate. The value V is a goodness measure of the polyhedral approximation to the true boundary contributed by this simplex. The minimum of all V's of the interior simplices is denoted by V_{min}, a measure for the overall goodness of the approximation. We focus the search on candidates satisfying the "visibility conditions" and eliminate the exterior simplices by deleting the candidate simplex with the least V among all the candidates, until the number of points on the boundary of the polyhedral approximation is the same as that of the original set of the boundary points and the elimination of this candidate could not further increase V_{min}.

We explore the properties of interior circumspheres versus those of exterior ones. When the distribution of boundary points respects the local geometries of the boundary they approximate, the envelope of all the circumspheres of interior simplices is close to the true boundary. Consider the maximum distance between the faces of a simplex that are on the boundary and its circumsphere: the smaller the distance is, the better the simplex approximates the boundary. The distance for an interior simplex is smaller than that of an exterior one. We use the inverse of this distance as the value of V for the simplex. Another possible candidate for V is the inverse of the circumsphere radius. Figure 3.7(a) illustrates that for smooth shapes the exterior triangles tend to be flat and have larger circumcircles, compared with the interior ones.

We reiterate that the condition on the distribution of the boundary points has to be checked with respect to the shape of a region, to ensure that the circumsphere heuristic rule for elimination works. For example, more points are needed to approximate the boundary of a region with finger-like narrow parts, as shown in Figure 3.7(b).

The sculpture method is stated as follows:

Algorithm 3.1 *Sculpture method [7]:*

1. Construct a set G of all simplices in the triangulation and a set H of candidate simplices satisfying "visibility conditions";
2. Order the simplices of H according to their V values;
3. Consider the simplex s of H having the least V. If the number of vertices of the polyhedron formed by the set G is less than that of the original set of points or the removal of s increases the V_{min} of H, then delete s from G and H; otherwise, return G and terminate;
4. Update neighbors n_i of s in G. Add n_i to H. Go to Step 2.

Visibility conditions:

- In 2D, simplices with one edge and two vertices on the boundaries.

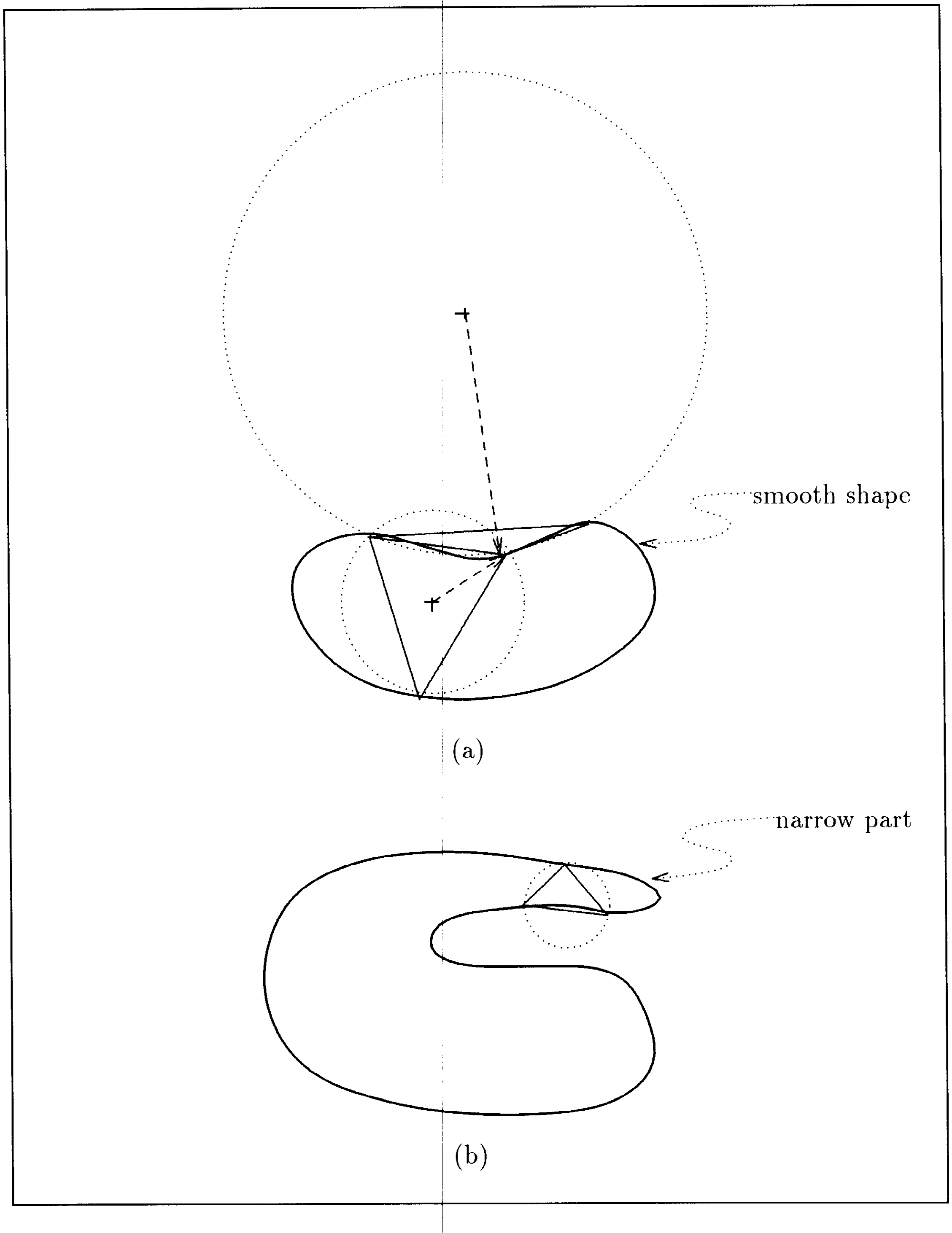

smooth shape
(a)
narrow part
(b)

Figure 3.7: Circumcircle properties of exterior triangles vs. interior ones: (a) a region with smooth shape; (b) a region with narrow-parts.

- In 3D, simplices with exactly one face, three edges, and three points on the boundary, or those with exactly two faces, five edges and four points on the boundary.

  ⋮

- In n-dimensions, simplices with one, or two, ..., up to $n-1$ proper faces on the boundary.

Each simplex s in the triangulation has a list of neighbors that share one or more common faces with s. When s is deleted from the triangulation, its neighbors need to be updated. A simplex satisfying the "visibility conditions" is added to H once and is deleted from H at most once in the elimination process. Let the number of simplices in the set G be M. When the set H is implemented as a heap, each insertion or deletion of H costs at most $O(\log M)$. The worst-case complexity of the algorithm therefore is $O(M \log M)$.

For systems whose stability regions have interleaved geometric structures or holes, the geometric extraction algorithm using the circumsphere heuristic described earlier often terminates before all the exterior simplices are eliminated. In this case, MAPS resorts to a more expensive method, the centroid method, that determines the membership of each simplex in the stability region: a simplex is classified as being in a stability region of an attractor if the trajectory emanating from the centroid of the simplex approaches the attractor in the limit or enters another simplex already in the stability region. For example, the buckling column discussed in Chapter 2 exhibits a banded phase space (see Figures 2.1 and 2.4). The circumsphere heuristic method terminates before all the boundary trajectory points are on the boundary of the polyhedral approximation. The centroid method is required to eliminate the remaining triangles.

The centroid method is implemented by an unraveling algorithm, adapted from the cell-to-cell algorithm [22], which links together simplices according to the mapping of their centroids under trajectory flows.

Algorithm 3.2 *Centroid method:*

1. Let α be the attractor under consideration, A be the set of simplices containing α in the interior or on the boundaries, G be the set of all simplices in the triangulation except for those of A, and C be a set initially empty;

2. For a simplex $s \in G$, let p be the next simplex that the trajectory starting at the centroid of s enters. Delete s from G. If $p \in A$, add s and all the reachable simplices[5] from s to A; otherwise, add s to C and make a pointer from p to s;
3. If G is empty, return A and terminate; otherwise, go to Step 2.

The time complexity of centroid method scales linearly with the number of simplices in the triangulation. The method requires integration over a time interval only once for each simplex.

The sculpture method and the centroid method work together to extract the geometric shape of the stability regions. The information about the boundary is then used in constructing the flow-pipe representation for the phase space. More specifically, the boundary information is used in labeling the simplices with flow directions at the boundary. Details of this will be discussed next.

3.3.3 Modeling trajectory flow pipes

The flow pipes are further geometric and dynamical characterizations of stability regions. They form the foundation for the search algorithm for synthesizing global control paths.

The polyhedral modeling of stability regions provides a geometric structure for reasoning about the stability boundaries. However, it does not provide information on how the trajectories flow within the region—the transient behavior very important in high-quality control design. The characterization of trajectory flows captures the missing information about the underlying vector-field flows; such information is essential for efficiently "looking for" good control reference trajectories.

We introduce the technique of flow pipes for describing the direction and the shape of the trajectory flows. The flow pipes form a discrete representation for phase space and yet preserve the fine geometries of the trajectories. The flow-pipe modeling, together with that of equilibria and stability regions, characterizes the transient and asymptotic behaviors of the dynamical systems in terms of the qualitative phase-space structures. Figure 3.8(a) shows a portion of the phase space for the Lienard equation $x'' + 0.5x' + x^2 = 1$, consisting of all trajectories ending at the attractor denoted by the symbol $+$. A flow pipe groups this collection of trajectories that exhibit the same qualitative feature into an equivalence class, as shown in Figure 3.8(b).

---

[5] A simplex x is reachable from s if there is a pointer or a chain of pointers from s to x.

Definition 3.3 flow pipe: *A flow pipe models a collection of trajectories exhibiting the same qualitative features in the phase space. It is an equivalence class of trajectory paths, each of which can be continuously deformed to another one, delimited by the boundaries of stability regions, trajectories connecting equilibria, or boundary surfaces dictated by applications*[6].

Constructing flow pipes

Computationally, a flow pipe is constructed by aggregating consistent geometric pieces with respect to the way that the flow travels from piece to piece. The catalogue of flow types on boundaries of the geometric pieces is a set of consistency constraints derived from the underlying dynamics of the vector field.

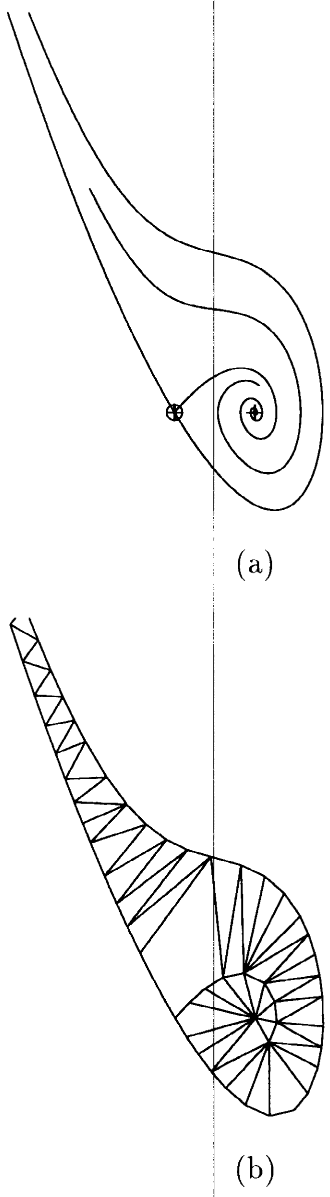
Given the simplicial tessellation of the phase space constructed by MAPS, each proper face f of a simplex s in the tessellation is classified into one of the three types with respect to the direction of the flow (see Figure 3.9), according to the following labeling scheme:

1. unidirectional flow:
   label *in:* Trajectories flow into s on f;
   label *out:* Trajectories flow out of s on f.
2. bidirectional flow:
   label *in-and-out:* Trajectories flow into s on some portion of f and flow out of s on other portion.
3. zero flow:
   label *tangent:* The flow does not intersect f. Hence the proper face f is on the boundary of the flow.

To determine the direction of the flow on a face of a simplex, the program samples N points evenly distributed on the face and computes the components of

---

[6] Formally, a flow pipe models a homotopy equivalence class of trajectory paths delimited by the boundaries of stability regions, trajectories connecting equilibria, or application-specific boundaries. If the ends of a flow pipe are squeezed to points, the flow pipe is a path homotopy equivalence class of trajectories.



(a)

(b)

Figure 3.8: Grouping trajectories into flow pipes: (a) a collection of trajectories with the same qualitative feature; (b) a corresponding flow pipe.

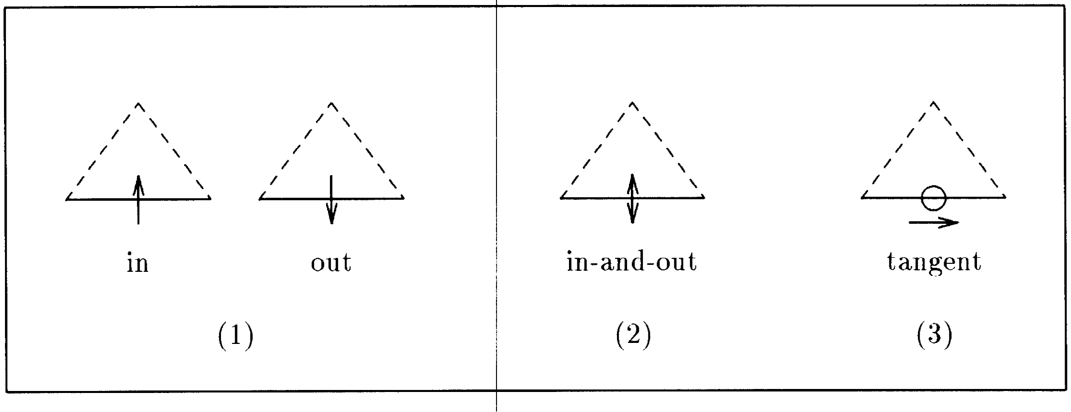


Figure 3.9: Classifying faces of simplices with respect to flows.

flow vectors along the normals of the face at these points. The stability boundaries necessarily separate flows and hence are on the flow-pipe boundary. Since the true stability boundary is only approximated by polyhedral faces, the above sampling method could yield incorrect flow labels on the boundary. The boundary information obtained in the characterization of the stability regions described earlier is used in labeling these zero flow boundary faces. In addition, the trajectories that connect saddles on the boundary and the attractor also separate trajectory flows. The simplicial tessellation is refined with the introduction of trajectory points sampled along these connecting trajectories.

A proper face of a simplex is called monotonic with respect to both the flow and the simplex, if the face is labeled either type 1 or type 3 in the above labeling scheme. A simplex is monotonic with respect to the flow if all of its proper faces are monotonic. Similarly, a polyhedron is monotonic if all of its proper faces are monotonic. Given two simplices agreeing on a non-monotonic proper face, the common non-monotonic face is canceled in the polyhedron formed by the two simplices; see Figure 3.10(a). A non-monotonic polyhedron continues to grow in this way until all of its proper faces are monotonic. The monotonic polyhedra are then aggregated to form flow pipes in such a way as to conserve the flow on their proper faces; see Figure 3.10(b). The following algorithm groups simplices into smallest monotonic polyhedra and clusters the monotonic polyhedra into flow pipes.

Algorithm 3.3 *Flow-pipe construction:*

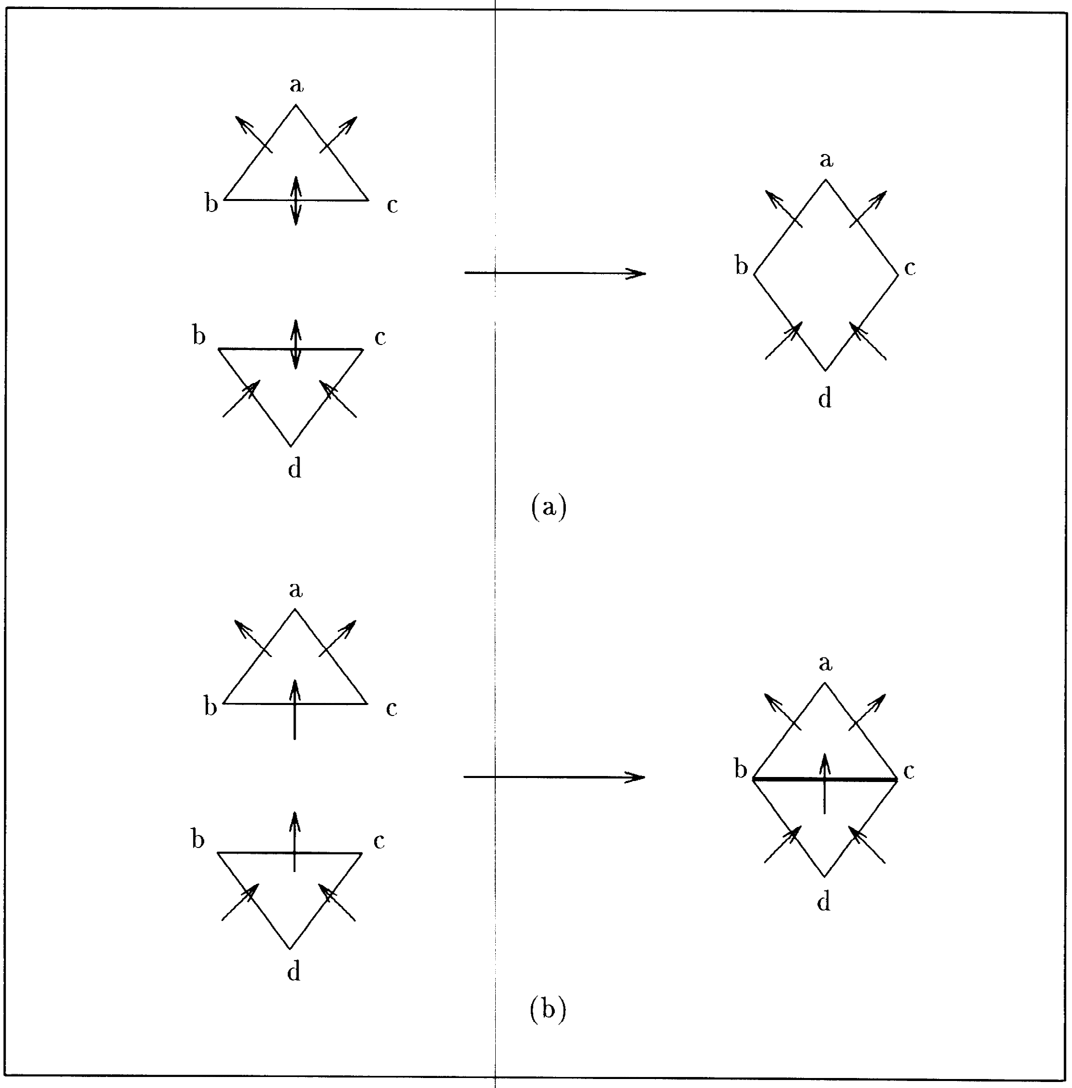

a
b
c
b
c
d
a
b
c
d
(a)
a
b
c
b
c
d
a
b
c
d
(b)

Figure 3.10: Construction of flow pipes: (a) grouping simplices to form a monotonic polyhedron by canceling common non-monotonic faces; (b) aggregating monotonic polyhedra to form a flow pipe according to flow directions at the boundaries.

1. Label proper faces of each simplex with respect to the flow.
2. Cluster simplices into monotonic polyhedra:
   Cluster simplices into equivalence classes with the equivalence relation on the proper faces of the simplices: two simplices are equal if they agree on a proper face with the same flow label: *in-and-out*.
3. Cluster monotonic polyhedra into flow pipes:
   Cluster monotonic polyhedra into equivalence classes with the equivalence relation on the proper faces of the polyhedra: two monotonic polyhedra are equal if they agree on a proper face with consistent flow directions: *in* and *out*, respectively.
4. Order the monotonic polyhedra in each class from Step 3 according to the flow direction. Return.

The resulting flow pipes model the trajectory flows. The boundary of a flow pipe is formed by the proper faces labeled *tangent* of simplices in the pipe.

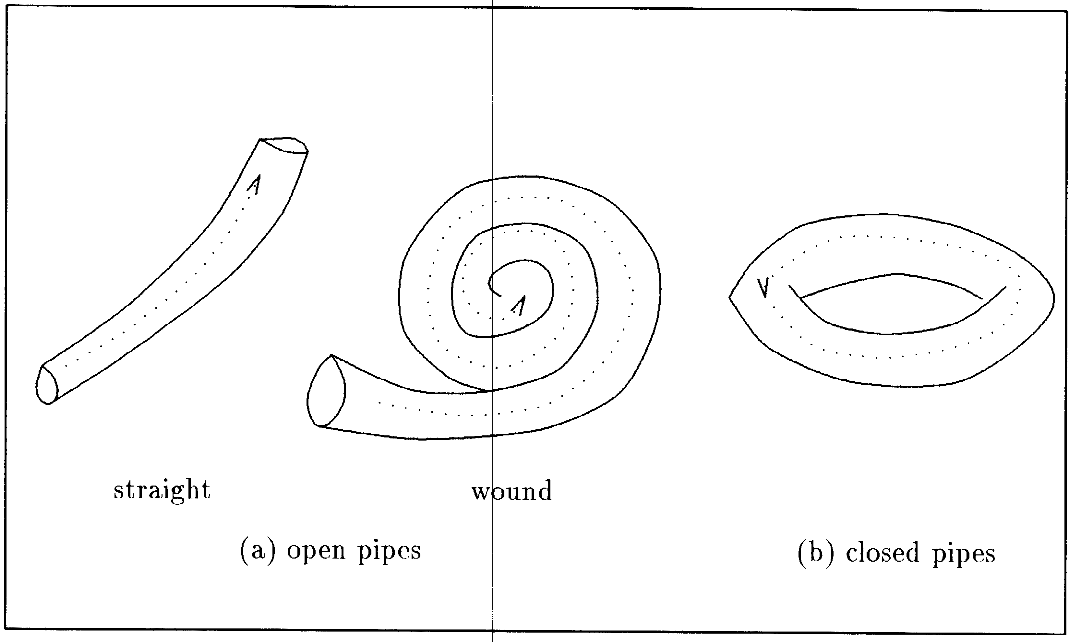
Classifying shapes of flow pipes

Flow pipes capture *dynamical shapes*. A flow pipe ends at an attractor and/or starts from a repellor. Saddles can not be in a flow pipe except for the boundary of the pipe, since flow pipes further decompose the stability regions. In a bounded region, each flow pipe either ends at an attractor or leaves the region.

Flow pipes exhibit *topological shapes*. Each flow pipe is classified according to its relative shape. A flow pipe around a stable limit cycle is a closed pipe. Most pipes are open. The open pipes can be relatively straight or highly wound. One, two, or more pipes can side by side wind into a spiral, called n-winding pipes. See Figure 3.11. We classify the pipes into the following categories:

1. open pipes: (a) straight pipes; (b) wound pipes.
2. closed pipes.

In the earlier Figure 3.8(b), for example, a single pipe winds into a spiral shape at its end and forms the stability region for the attractor. More complicated patterns can occur with more pipes or in higher dimensions.



(a) open pipes

(b) closed pipes

Figure 3.11: Topological shapes of flow pipes.

3.4 The MAPS Analysis Algorithm

3.4.1 The algorithm

We present the following algorithm for analyzing, extracting, and representing qualitative features of a dynamical system of any order in the phase space.

Algorithm 3.4 *MAPS analysis:*

(1) **Identify qualitative behaviors:**
  (a) locate equilibrium points/limit cycles and classify their stability types;
  (b) compute stable and unstable trajectories for each saddle/limit cycle;
  (c) identify those saddles/limit cycles whose unstable trajectories approach an attractor;
  (d) the stability boundary for the attractor is the union of the stable trajectories of those saddles/limit cycles identified in Step (c);
  (e) check if consistency rules are violated. If yes, look for missing equilibrium points/limit cycles and go to Step (a). Otherwise, go to the next step.

(2) **Extract geometric structures:**

(a) for each attractor, collect stability boundary points;
(b) tessellate the convex hull of the boundary points with a triangulation;
(c) extract a polyhedral approximation to the stability region.

(3) **Construct flow pipe:**
(a) refine the triangulation of the stability regions;
(b) label simplices with flow directions at boundaries;
(c) aggregate the labeled simplices into flow pipes.

(4) **Summarize qualitative behaviors and generate a high-level description:**
(a) compile the phase-space data structure from Step 1 into a relational graph;
(b) augment the graph with the geometric structures from Steps 2 and 3;
(c) report the graph as the output.

The set of consistency rules specify the conditions for the stability boundaries and are used in the algorithm to automatically locate missing saddles.

1. *The Existence Rule*: Every stability region of an attractor has a boundary in a phase space with multiple attractors;
2. *The Separation Rule*: Separatrices either form a closed surface or become unbounded on all ends.

The first rule states the existence of stability boundaries in a phase space with multiple attractors. The second rule describes the separation property of multiple stability regions. The separatrices are stability boundaries that separate two stability regions.

The first step of our algorithm is based on a numerical method proposed by Parker and Chua [38] for numerically determining stability boundaries of planar systems. We have augmented their method with the set of consistency rules they suggested to automate the locating of saddles. Since the Newton-Raphson method [41] used in finding equilibrium points requires an initial guess, the Parker-Chua method uses a grid to set up initial guesses and is able to find all the stable and unstable equilibrium points under normal circumstances. However, they require that the initial guesses for saddles be provided manually by the user. We seek to automate saddle locating by focusing the search for missing saddles on the most likely places using partial boundary information already obtained, or by refining the initial guesses for the Newton-Raphson method. We want to emphasize

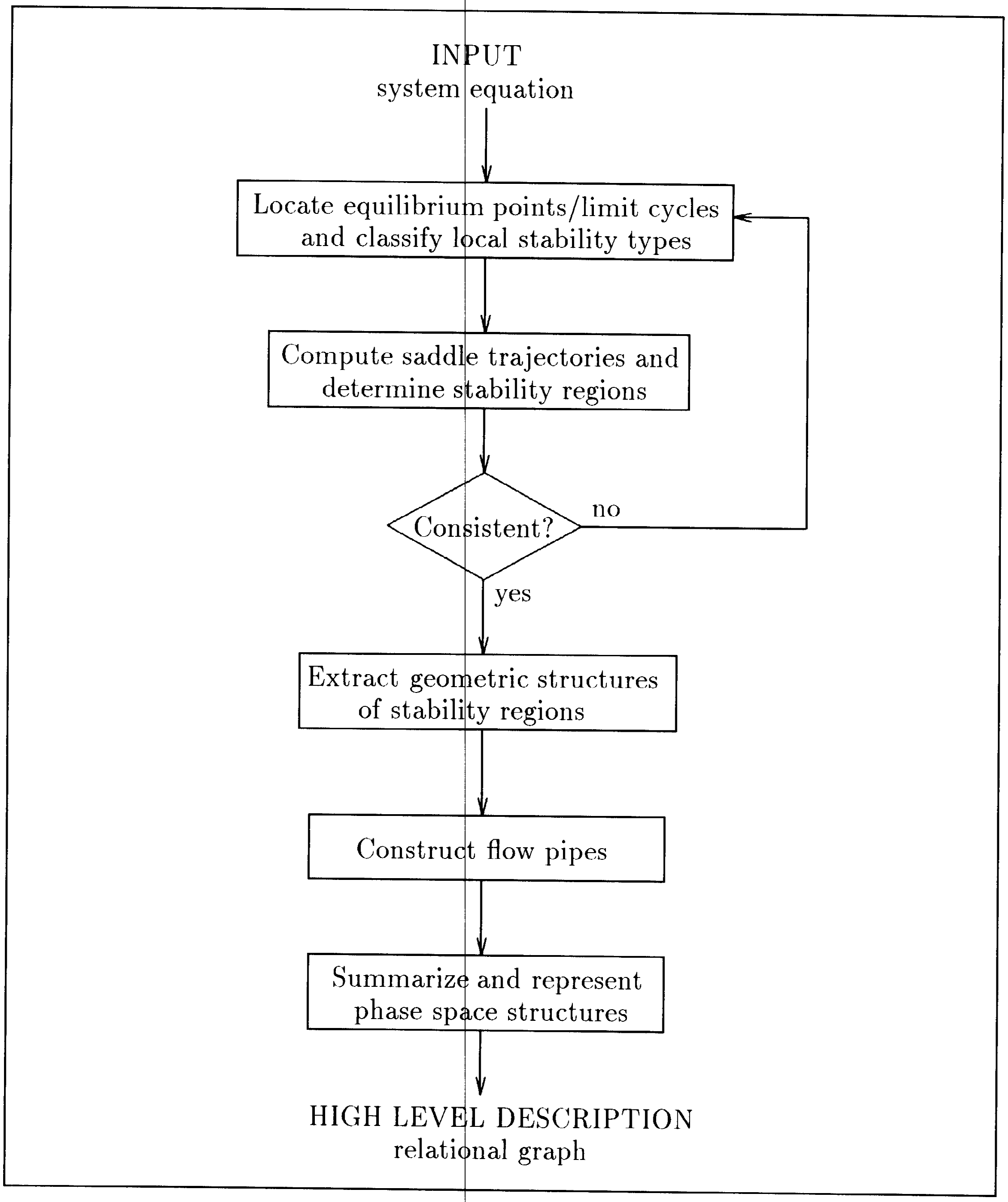

INPUT
system equation
Locate equilibrium points/limit cycles and classify local stability types
Compute saddle trajectories and determine stability regions
Consistent?
no
yes
Extract geometric structures of stability regions
Construct flow pipes
Summarize and represent phase space structures
HIGH LEVEL DESCRIPTION
relational graph

Figure 3.12: The flow chart of MAPS.

that our algorithm is valid for higher dimensional systems as well and generates a symbolic description of the phase-space structure. The Parker-Chua method is designed for numerically analyzing planar systems only. We note that our algorithm also finds degenerate boundaries if there are any.

In the second step, the algorithm extracts the geometric structures of stability regions. It uses a triangulation method to tessellate the phase space of the system and computes polyhedral approximations to the regions. MAPS internally represents a trajectory with a sequence of integration points. It uses an adaptive 4th-order Runge-Kutta method [41] to integrate trajectories and to generate the boundary points that approximate a stability boundary.

The third step refines the triangulation from the second step and constructs flow pipes, using the information about the stability boundaries. The unstable trajectories of saddles connect the saddles with the attractors. We call these trajectories the connecting trajectories. The triangulation from Step 2 is refined by the insertion of trajectory points from the connecting trajectories. The algorithm for the flow pipes clusters simplices into equivalence classes as described in Section 3.3.3.

The last step represents the data structure of the phase-space geometries and topologies with a relational graph. The details of each step will be illustrated with an example in the following section.

The flow chart of MAPS is shown in Figure 3.12. The input to MAPS is a system of governing equations for a dynamical system. We could also start with a set of measured states from experiments. These numerical states could then be analyzed and clustered to form phase portraits. The phase space could also be reconstructed from a measured time series of just one state variable, using a technique called delay coordinates introduced in [36, 48].

3.4.2 The main illustration

We illustrate how MAPS computes the high-level description of a dynamical system with an example. Consider a 2nd-order nonlinear system

$$\begin{cases} x' = -3x + 4x^2 - xy/2 - x^3 \\ y' = -2.1y + xy + u \end{cases} \tag{3.1}$$

where u is a parameter. For the parameter value $u = 0.2$, the vector field within the region $-1.0 \leq x \leq 4.0$ and $-1.0 \leq y \leq 4.0$ is shown in Figure 3.13. MAPS

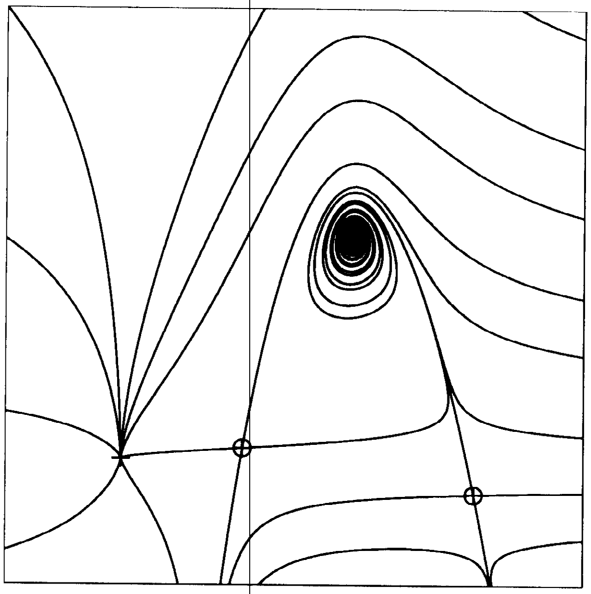


Figure 3.13: The main illustration: the vector field of a 2nd-order nonlinear system.

analyzes the qualitative behavior of the system within this region. The program actually performs the analysis in a slightly larger bounding box to account for trajectories that enter the bounding box again after a short exit. (The current implementation magnifies the original bounding box by a factor of 1.5 on each side.) The equilibrium points of the system are found by a zero-finding method on $f(x, u)$—the Newton-Raphson method. MAPS locates four equilibrium points within the region and classifies their stabilities by inspecting the eigenvalues of Jacobians at the equilibrium points: two attractors at $(0.0, 0.0952)$ and $(2.0, 2.0)$ and two saddles at $(1.05, 0.19)$ and $(3.05, -0.21)$, all shown in Figure 3.14(a) (Note that the attractors are represented with the symbol $+$ and the saddles are represented with the symbol $\oplus$.). The stable and unstable trajectories of the saddle are then computed by integrating the system from a small neighborhood of the saddle in the directions of the stable and unstable eigenvectors backwards and forwards, respectively.

Since one of the unstable trajectories of each saddle goes to the attractor at $(2.0, 2.0)$, the stability boundary of the attractor consists of the stable trajectories of both saddles. Similarly, the stability boundary of the attractor at $(0.0, 0.0952)$

consists of the stable trajectories of the saddle at $(1.05, 0.19)$, one of whose unstable trajectories goes to the attractor. However, within the region of interest there are trajectories that leave the bounding box. These trajectories can be conveniently thought of as the stable trajectories of an attractor at infinity. Therefore, the stable trajectories of the saddle at $(3.05, -0.21)$ form the stability boundary for the attractor at infinity, for one of the unstable trajectories of the saddle leaves the bounding box. Consistency rules are checked and satisfied. At the end of this step, MAPS finds three qualitatively different regions associated with the three attractors and internally represents the phase-space structure in a data structure: the attractors are connected with each other via saddles and associated with stability boundaries (Figure 3.14(b)).

The second step extracts a polyhedral approximation to each stability region preserving the gross features of the shape of the region. Consider the stability region of the attractor at $(2.0, 2.0)$. The stability boundary is numerically approximated by a collection of trajectory points (70 in total for this example) relatively uniform and dense on the boundary; see Figure 3.15(c).

A Delaunay triangulation is performed on this set of points. As the result, the convex hull of the points is tessellated with triangles; see Figure 3.15(d). Since the boundary points are distributed reasonably densely and uniformly on the boundary, the polyhedral shape of the stability region is contained in the triangulation of the convex hull. MAPS successfully eliminates the triangles exterior to the polyhedron with the heuristic method, without resorting to the centroid method. The resulting region is shown in Figure 3.16(e).

MAPS then computes the flow-pipe modeling of the phase space. The triangulation of the stability region obtained above is further refined with additional points from connecting trajectories. MAPS labels each triangle in the refined triangulation with flow direction on its faces and clusters these triangles into equivalence classes. This results in two flow pipes that form the stability region from the second step (Figure 3.16(f)).

MAPS compiles the data structure for equilibria, stability regions, and flow pipes from earlier steps into a relational graph; see Figure 3.17. It reports its findings to the user in a symbolic summary in Figure 3.18.

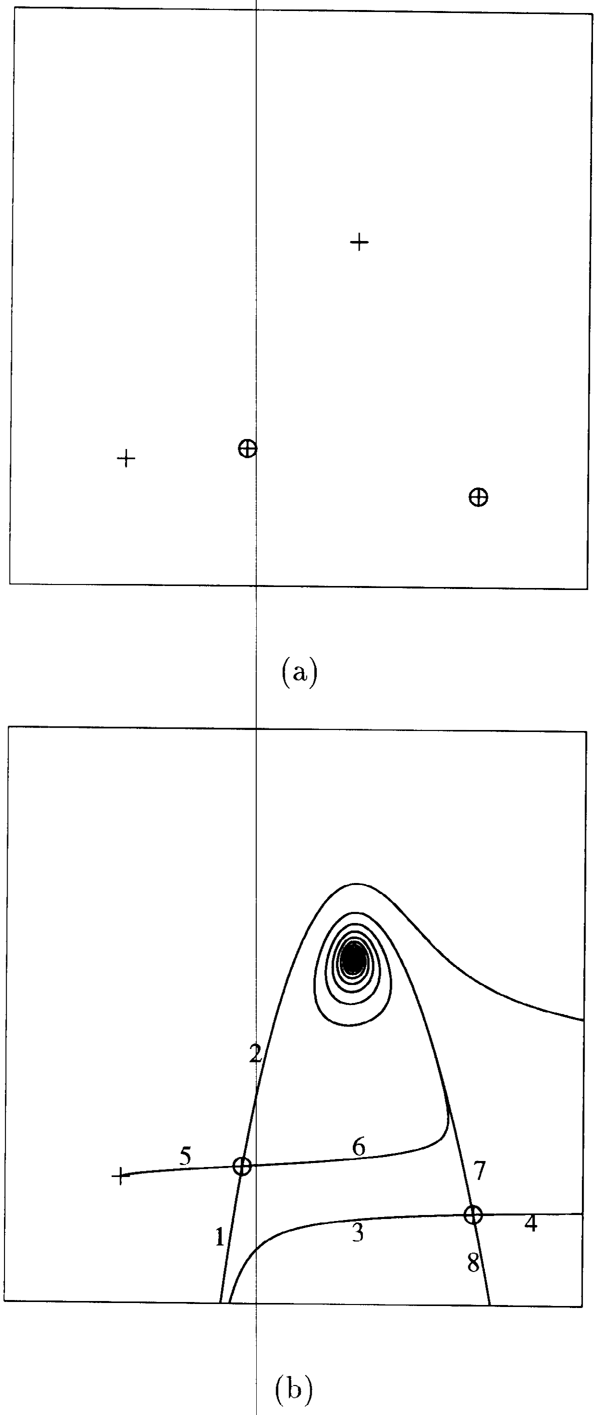
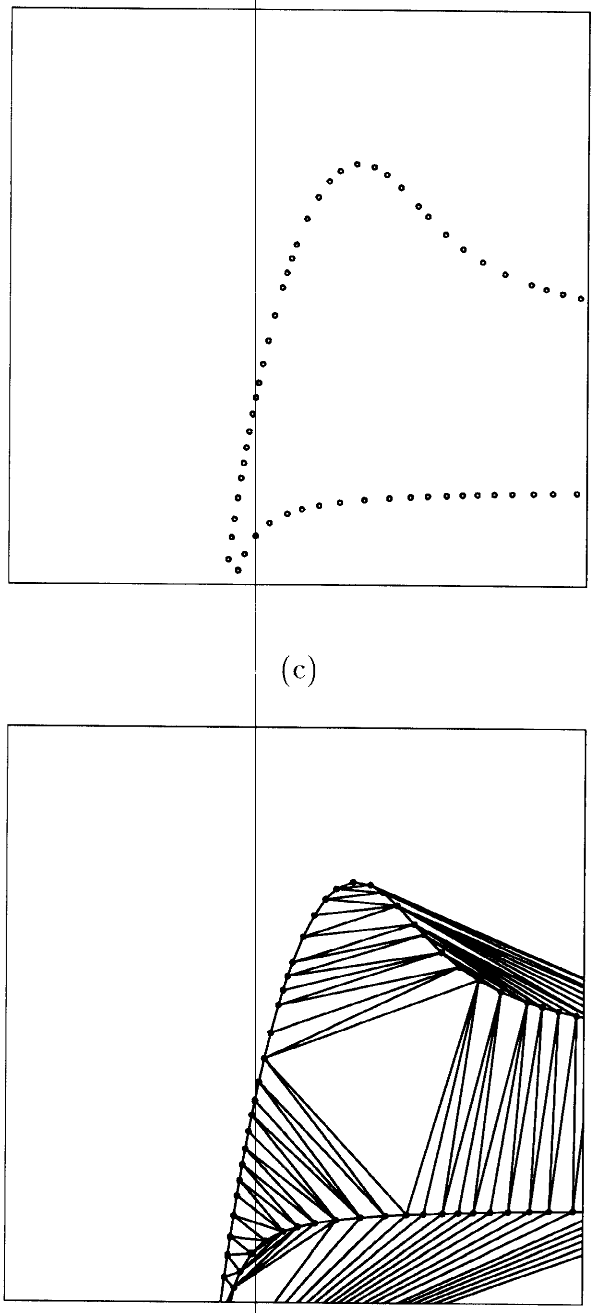


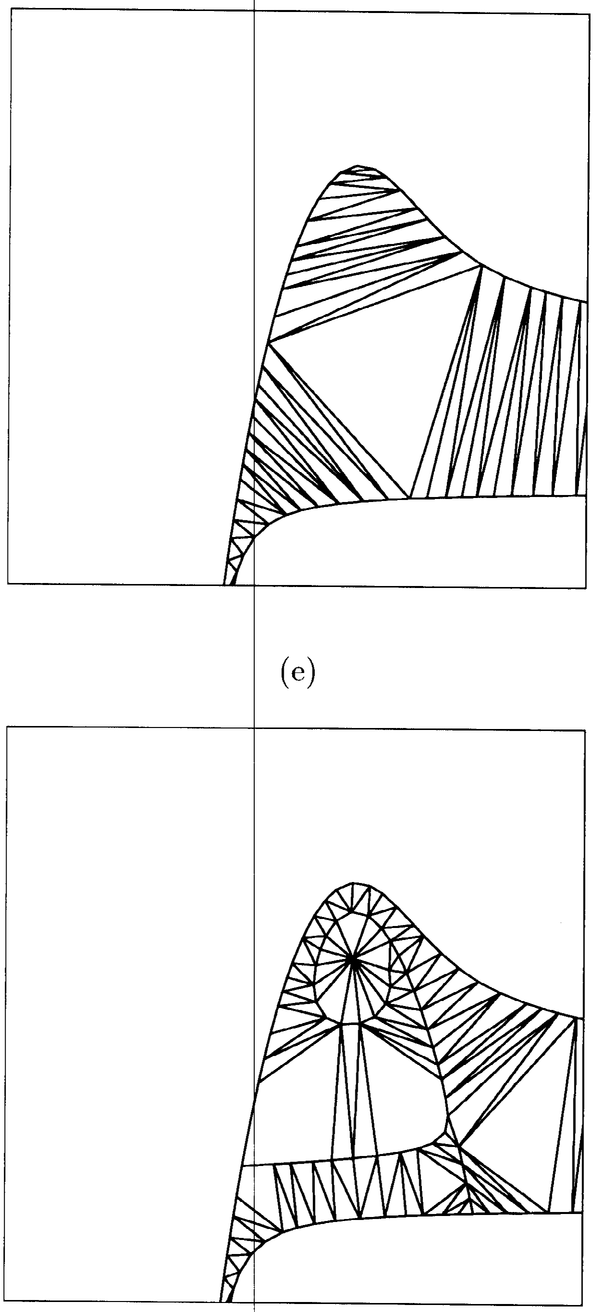
Figure 3.14: The main illustration—MAPS output: (a) equilibrium points; (b) boundary and connecting trajectories. (to be continued)



(c)

(d)

Figure 3.15: (cont'd) The main illustration—MAPS output: (c) points on the stability boundary for one of the attractors; (d) triangulation of the convex hull.



(e)

(f)

Figure 3.16: (cont'd) The main illustration—MAPS output: (e) polyhedral approximation to the stability region computed from the triangulation on boundary points; (f) two flow pipes computed from the refined triangulation. The flow pipes form the stability region.

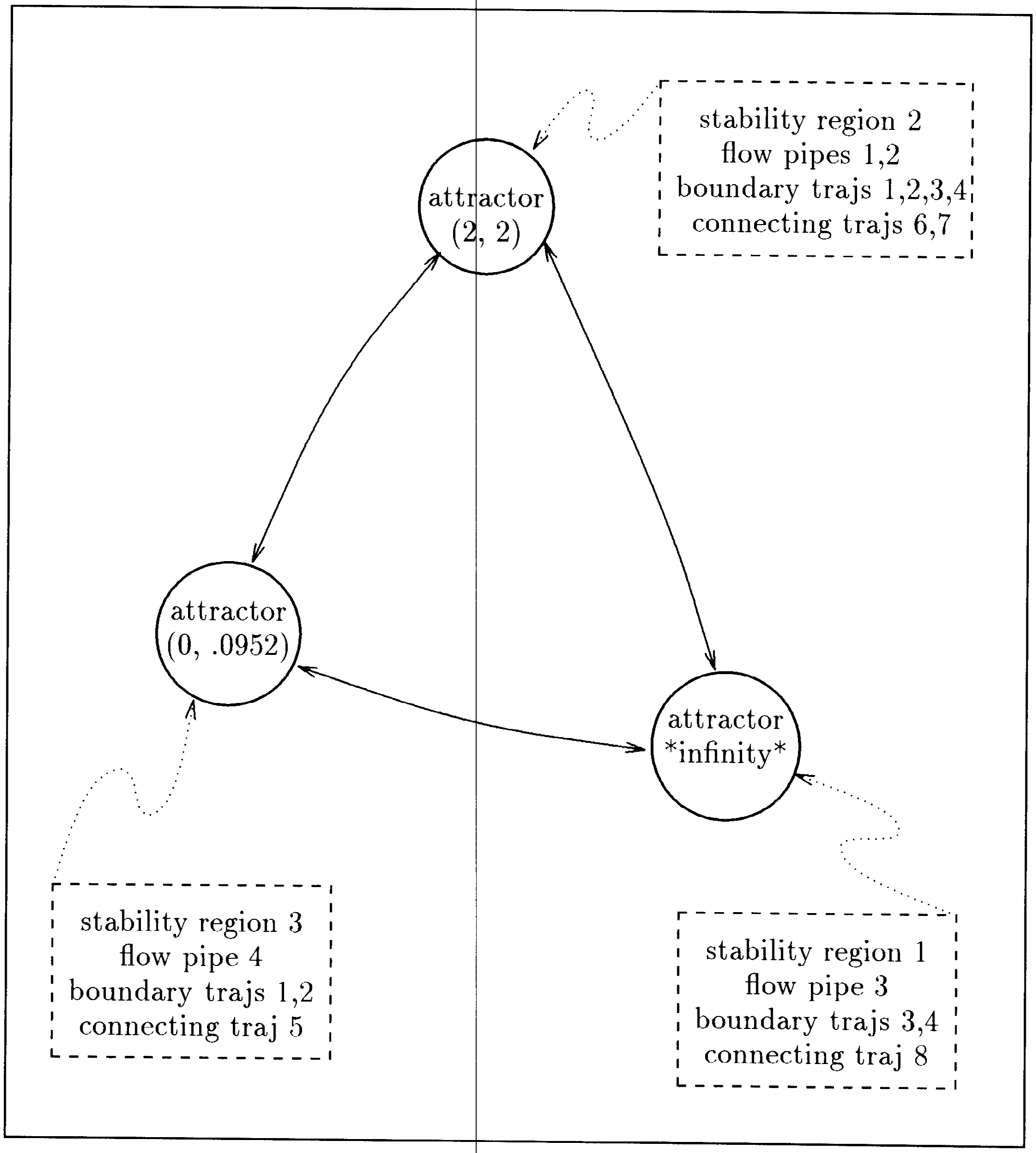

stability region 2
flow pipes 1,2
boundary trajs 1,2,3,4
connecting trajs 6,7
attractor
(2, 2)
attractor
(0, .0952)
attractor
infinity
stability region 3
flow pipe 4
boundary trajs 1,2
connecting traj 5
stability region 1
flow pipe 3
boundary trajs 3,4
connecting traj 8

Figure 3.17: The main illustration: the relational graph constructed by MAPS.

```
<equilibrium-points:
 equilibrium 1. (saddle at (3.05 -.21))
 equilibrium 2. (attractor at (2. 2.))
 equilibrium 3. (saddle at (1.05 .19))
 equilibrium 4. (attractor at (0. .0952))>

<trajectories:
 <boundary-trajectories:
  trajectory 1. (from *infinity* to (1.05 .19))
  trajectory 2. (from *infinity* to (1.05 .19))
  trajectory 3. (from *infinity* to (3.05 -.21))
  trajectory 4. (from *infinity* to (3.05 -.21))>
 <connecting-trajectories:
  trajectory 5. (from (1.05 .19) to (0. .0952))
  trajectory 6. (from (1.05 .19) to (2. 2.))
  trajectory 7. (from (3.05 -.21) to (2. 2.))
  trajectory 8. (from (3.05 -.21) to *infinity*)>>

<stability-regions:
 stability-region 1.
   attractor at *infinity*
   stability-boundary: (trajectory 4. trajectory 3.)
   connecting-trajectories: (trajectory 8)
 stability-region 2.
   attractor at (2. 2.)
   stability-boundary: (trajectory 4. trajectory 3. trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 7 trajectory 6)
 stability-region 3.
   attractor at (0. .0952)
   stability-boundary: (trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 5)>

<flow-pipes:
 flow-pipe 1. (from *infinity* to (2. 2.))
   boundary: (trajectory 3. trajectory 1. trajectory 7. trajectory 6.)
 flow-pipe 2. (from *infinity* to (2. 2.))
   boundary: (trajectory 4. trajectory 2. trajectory 7. trajectory 6.)
 flow-pipe 3. (from *infinity* to *infinity*)
   boundary: (trajectory 4. trajectory 3. trajectory 8.)
 flow-pipe 4. (from *infinity* to (0. .0952))
   boundary: (trajectory 2. trajectory 1. trajectory 5.)>
```

Figure 3.18: The main illustration—MAPS output: a symbolic summary.

3.4.3 Implementation details

MAPS is implemented in the Scheme Language. Internally, a point in phase space is a vector of the coordinates. An equilibrium point is a list of its position, eigenvalues and eigenvectors of Jacobian at the point, and its stability type. A trajectory specifies the type of the trajectory, start and end points, and a sequence of integration points as a Scheme stream object. The type could be either `stable-man` or `unstable-man`, standing for stable manifold or unstable manifold; a trajectory typed `stable-man` is integrated backwards, and a trajectory typed `unstable-man` is integrated forwards. A stability region is represented as a list of the attractor, the boundary and connecting trajectories, and the polyhedral approximation. A flow pipe is a list of its polyhedral approximation, the boundary, and the start and end faces. Table 3.1 summarizes the internal representation of these objects. The examples for the stability region and the flow pipe are omitted due to space limitations.

3.4.4 More examples

Our algorithm also applies to higher-dimensional systems. Consider the following 3rd-order nonlinear system

$$\begin{cases} x' = y \\ y' = z - x^2 - y \\ z' = 1.0 - y^2 - 0.8z^2 \end{cases}$$

MAPS locates an attractor at $(1.06, 0.0, 1.12)$ and a saddle at $(-1.06, 0.0, 1.12)$ within the region $-5.0 \leq x \leq 5.0$, $-5.0 \leq y \leq 5.0$, and $-5.0 \leq z \leq 5.0$, and determines that the stable trajectories of the saddle form the stability boundary for the attractor. The stability boundary is a two-dimensional surface and is approximated by a set of relatively evenly spaced trajectories. MAPS then tessellates the phase space with tetrahedra and extracts a polyhedral approximation to the stability region of the attractor (see Figure 3.19). As described earlier, MAPS uses general geometric primitives — the n-simplices — to approximate stability regions in n dimensions.

We have also run MAPS on a system that is not generic and falls outside the domain of our method (see discussion in Section 3.5.1). The system is the earlier 2nd-order example (3.1) in Section 3.4.2 with parameter value $u = 0$. MAPS

Object	*a point*
Representation	a vector of coordinates
Example	`#(2.  2.)`
Object	*an equilibrium*
Representation	a list: (point, eigenvalues/eigenvectors, stability type)
Example	`(#(2.  2.)  (-.05+1.41i -.05-1.41i)` `(#(1.  0) #(.025 1.))  stable)`
Object	*a trajectory*
Representation	a list: (type, start, end, integration points)
Example	`(stable-man *infinity* #(3.05 -.21)` `(#(3.05 -.21) .  #[promise 26]))`
Object	*a stability region*
Representation	a list: (attractor, boundary trajectories, connecting trajectories, polyhedron)
Example	...
Object	*a flow pipe*
Representation	a list: (polyhedron, boundary, start faces, end faces)
Example	...

Table 3.1: The internal representation of phase-space data objects.

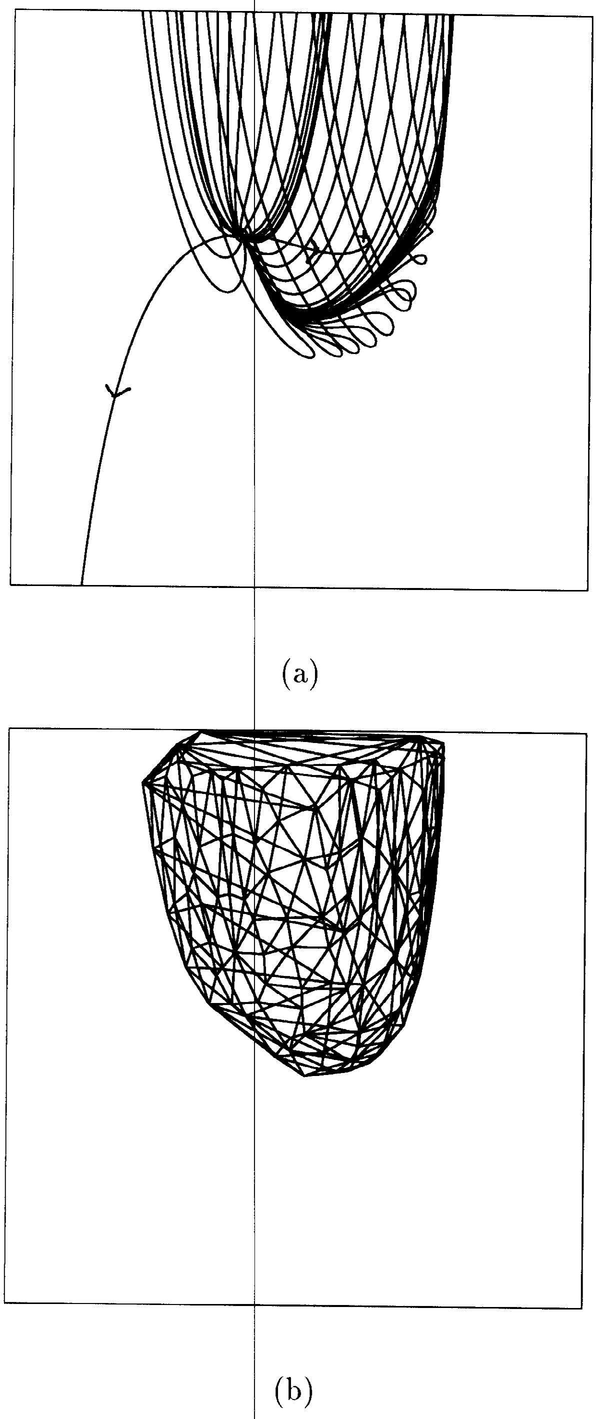


Figure 3.19: The analysis of a 3rd-order nonlinear system: (a) projection of stability boundary and connecting trajectories in x-z plane; (b) projection of polyhedral approximation in x-z plane.

locates four equilibrium points within the region $-1.0 \leq x \leq 4.0$ and $-1.0 \leq y \leq 4.0$ and computes the stable and unstable trajectories of saddles. It terminates with the following partial description:

```
<equilibrium-points:
 equilibrium 1. (saddle at (3. 0.))
 equilibrium 2. (attractor at (2.1 1.98))
 equilibrium 3. (saddle at (1. 0.))
 equilibrium 4. (attractor at (0. 0.))>

<trajectories:
  trajectory 1. (from *infinity* to (1. 0.))
  trajectory 2. (from *infinity* to (1. 0.))
  trajectory 3. (from (1. 0.) to (3. 0.))
  trajectory 4. (from *infinity* to (3. 0.))
  trajectory 5. (from (1. 0.) to (0. 0.))
  trajectory 6. (from (3. 0.) to (2.1 1.98))
  trajectory 7. (from (3. 0.) to *infinity*)>

<saddle-connection:
  trajectory 3. (from (1. 0.) to (3. 0.))>
```

MAPS discovers a saddle connection in the course of determining the relational graph of the phase-space structure: the trajectory that serves both as an unstable trajectory of the saddle at $(1.0, 0.0)$ and as a stable trajectory of the saddle at $(3.0, 0.0)$. The saddle connection is labeled as trajectory 3 in Figure 3.20. MAPS concludes that the system is not generic—which also implies structural instability for the planar system here—and abandons any further efforts to characterize the phase-space structure. Since saddle connections are often precursors to chaos or structural instability, we believe that they are important in partially characterizing the phase-space structures of chaotic or structurally unstable systems.

3.4.5 Hierarchical extraction and representation of phase-space information

We have described and demonstrated our algorithm for analyzing a dynamical system through successive computations on the system, starting from its system equation representation. MAPS generates a high-level description of the dynamical system at the end of the analysis. To bridge the large semantic gap between

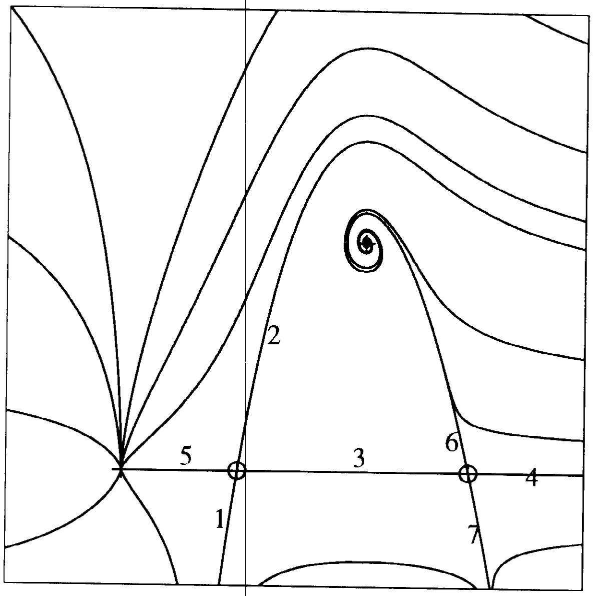


Figure 3.20: A 2nd-order system with a saddle connection.

the deduced symbolic description and the system equation representation of the input, MAPS has employed multiple intermediate representations for the dynamical system, shown in Figure 3.21.

MAPS extracts the information incrementally, applying a set of operations to each intermediate representation. At each level of the representation, implicit properties, such as spatial relations, of the system at different scales are made explicit and thus can be accessed and manipulated by the operators at the next level of the representation. In the order of analysis, MAPS first generates a local description of equilibrium points, limit cycles, and their eigenstructures. It then computes the polyhedral approximations to stability regions and trajectory flows. Finally, it pieces together the local information about each limit set to form a global picture of the phase-space structure: the relational graph describing the interactions of equilibrium points, limit cycles, stability regions, and flow pipes.

The *internal representation* of the phase-space description as a relational graph captures the qualitative aspects of the phase-space structure. In the relational graph, nodes are attractors and arcs denote the relations between their stability regions. Each node has information about the attractor it represents, the associated stability region, trajectory flows, and their polyhedral approximations, and

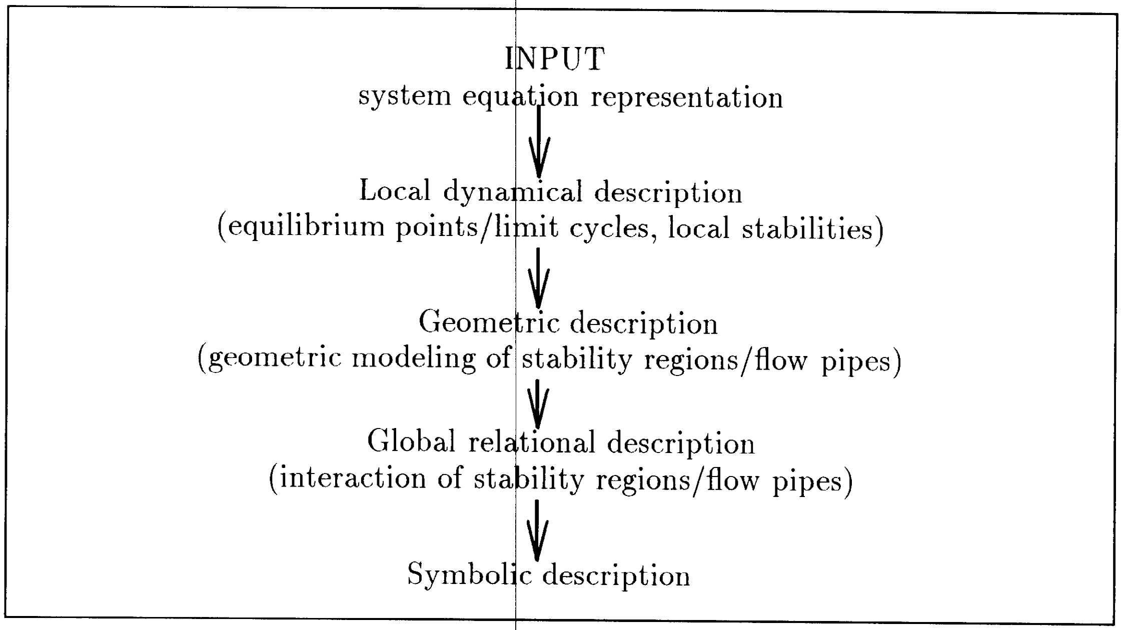

INPUT
system equation representation
Local dynamical description
(equilibrium points/limit cycles, local stabilities)
Geometric description
(geometric modeling of stability regions/flow pipes)
Global relational description
(interaction of stability regions/flow pipes)
Symbolic description

Figure 3.21: A multi-layered representation for a dynamical system used in MAPS.

the boundary trajectories and boundary equilibrium points and limit cycles. Each equilibrium point has information about its position, eigenvalues and eigenvectors, and stability type.

3.5 Discussion

MAPS analyzes qualitative behaviors of nonlinear dynamical systems using knowledge about stability and trajectory flows from dynamical systems theory. It extracts the geometric information from the numerical results and represents it with a qualitative phase-space structure. In this section, we discuss the class of dynamical systems to which our method applies, further extensions to MAPS, and the role of MAPS in assisting control engineers in visualizing behaviors of dynamical systems.

3.5.1 Scope of the analysis

We have stated that the theoretical basis of our algorithm for locating stability boundaries is the result of Chiang et al. [10]. Chiang et al. give a topological and dynamical characterization of the stability boundaries for a class of autonomous dynamical systems and show that the stability boundary of an attractor is formed by the stable trajectories of equilibrium points and limit cycles on the boundary. This class of dynamical systems is defined by three conditions:

1. Hyperbolicity: All equilibrium points and limit cycles on stability boundaries are hyperbolic[7].
2. Transversality: The stable and unstable trajectories of equilibrium points and limit cycles on stability boundaries are transversal to each other.
3. Finite limits of boundary trajectories: Every trajectory on the stability boundaries approaches an equilibrium point or a limit cycle as $t \to \infty$.

Conditions (1) and (2) are generic properties satisfied by almost all dynamical systems. The example in Section 3.4.4 that has a saddle connection violates condition (2). Structurally stable systems satisfy the generic conditions (1) and (2) (see [19, 46] for a definition of generic properties and [18] for discussions on structural stability of dynamical systems). Since many systems encountered in engineering applications are structurally stable, we further restrict ourselves to structurally stable systems for the sake of simplicity[8]. Structural stability implies the absence of quasi-periodic orbits. Condition (3) excludes some structurally stable systems. Our method applies to a fairly large class of autonomous dynamical systems as defined by the three conditions. We note that a periodic non-autonomous dynamical system can be converted into an autonomous system [18] and can also be analyzed by MAPS.

In its current implementation, MAPS does not handle systems having chaotic attractors, in which stability regions and/or stability boundaries exhibit fractal structures. It also assumes that the number of equilibria and limit cycles of a system is finite.

---

[7] Consider the Jacobian of the vector field at an equilibrium point. If all the eigenvalues of the Jacobian have non-zero real parts, then the equilibrium point is *hyperbolic*. Similarly, a limit cycle is hyperbolic if none of its characteristic multipliers, the eigenvalues of its Poincaré map, lies on the unit circle in the complex plane.

[8] This is not to say that structurally unstable systems are not interesting. In fact, Hamiltonian systems are important in investigating phenomena in which energy dissipation is negligible.

3.5.2 Extensions

The theoretical basis of our approach holds regardless of phase-space dimensionality, as does the geometric extraction method of our algorithm. The polyhedral approximation is a natural, economical representation for the boundary surface. We have run MAPS on systems in dimensions up to three. In higher dimensions, the computation can be very expensive. We have noted in Section 3.3.2 that the complexity of the Delaunay triangulation grows exponentially with the number of dimensions. The number of points necessary to approximate the boundary is also a function of the quality of the approximation. The computational complexity reflected in managing the large data structure for representing phase-space geometries needs to be addressed at greater length in future research. One case in which a reduction of the complexity is possible is when the interesting dynamics is constrained to a submanifold of a high-dimensional phase portrait. A program could recognize such structure of the phase space and restrict the computation to that subspace.

In dimensions three or higher, a stability boundary surface consists of trajectories sweeping out the surface. Obtaining a set of relatively evenly spaced trajectories of the boundary to approximate the surface is challenging. MAPS currently uses spacings between adjacent boundary trajectories and local curvature to measure the quality of the approximation. Combining these measures with other kind of metric information about the boundary surface could improve the approximation. Much remains to be explored in order to best approximate complicated curved hypersurfaces and to reflect the underlying dynamics.

MAPS extracts the geometric structure using the information about the stability boundaries and additional information about the trajectories. Other complicated stability regions, such as those containing holes, can also be tackled with this approach. More work needs to be done to catalogue the extraction of region shapes with various kinds of approaches.

We have shown that MAPS detected a saddle connection in a structurally unstable system. A considerable amount of work needs to be directed toward exploring robust methods for detecting saddle connections and extending MAPS to recognize chaotic attractors. In the case when the stability boundary is fractal, although the fine structure of the boundary would be difficult to characterize with a crisp surface, it is still possible to envelop the fractal structure with a boundary layer of certain thickness. Another possible extension is to augment the current program with a bifurcation analysis, similar to Abelson's Bifurcation Interpreter [3].

3.5.3 The use of MAPS in visualization

The qualitative phase-space description can assist engineers in designing controllers for complex systems.

We envision that a control engineer uses MAPS to observe *qualitative changes* of phase-space structures when varying control parameters. These changes include:

1. the birth, disappearance, and movement of equilibrium points and limit cycles;
2. the enlargement and shrinking of stability regions;
3. other phase-space qualitative changes.

A program has been constructed for graphically displaying three-dimensional geometric structures: it renders the surfaces of 3D polyhedral structures with different lighting parameters. With such a graphics rendering tool, an engineer can visualize the geometries of a phase portrait, interactively edit the system to observe corresponding changes, and cut-and-paste useful portions of phase portraits with different parameter values to form a composite having the desired properties.

In the next chapter, we automate the above scenario with the Phase Space Navigator that autonomously synthesizes nonlinear controllers in phase spaces. More importantly, the method will be able to synthesize a nonlinear control system whose phase space is high dimensional and difficult to visualize, or on which the desired control properties are impossible to obtain with traditional design techniques. The *topological* and *dynamical* modeling of the phase-space stability regions and trajectory flows forms the basis for our method.

3.6 Summary of the Chapter

We have developed a qualitative method for automatically analyzing phase-space structures of nonlinear dynamical systems and have constructed MAPS to demonstrate the method. MAPS "looks" at a phase space, finds qualitatively different regions—the stability regions, models trajectory flows with equivalence classes, and extracts and represents the qualitative features. It employs deep domain knowledge of dynamical systems theory to recognize the qualitative structures of phase spaces. It computes a high-level description of a dynamical system through a combination of numerical, combinatorial, and geometric computations and represents the phase-space structure with a relational graph. The qualitative representation of dynamical systems and the computational formulation of the knowledge of engineering analysis enable us to build computer programs that automatically design high-quality controllers for nonlinear systems. In the next chapter we will use MAPS to analyze a collection of phase spaces each of which corresponds to fixed control parameter values.

Chapter 4

Automatic Phase-Space Control Synthesis — Phase Space Navigator

4.1 Introduction

This chapter develops Phase Space Navigator, an autonomous system for control synthesis of nonlinear dynamical systems. The Phase Space Navigator automatically designs a controller for a nonlinear system in phase space. It generates control laws by synthesizing the desired phase-space flow "shapes" for the system and planning and navigating the system along good control trajectories in phase space.

The Phase Space Navigator relies on the phase-space knowledge of dynamical control systems. It employs the MAPS program to extract and represent qualitative phase-space structures characterizing the qualitative aspects of the dynamics. The control synthesis utilizes flow pipes to model phase spaces and to search for global control paths. We will present the novel idea of phase-space navigation as a paradigm for high-performance nonlinear control design. Algorithms for control trajectory planning and tracking will be described. The synthesis method will be illustrated with an example of synthesizing anti-buckling control laws for a steel column.

4.2 Automatic Control Synthesis in Phase Space

The Phase Space Navigator synthesizes a control system from a geometric point of view in phase space. The control of a dynamical system is interpreted as the "steering" of the system trajectory, emanating from some initial state, to the desired state by a control signal.

4.2.1 Overview of the Phase Space Navigator

The Phase Space Navigator consists of a global control path planner, a local trajectory generator, and a reference trajectory follower. The global path planner finds optimal paths from an initial state to the goal state in phase space, consisting of a sequence of path segments connected at intermediate points where the control parameter changes. A brute-force, fine-grain search in high-dimensional phase spaces would be prohibitively expensive. High-level descriptions of the phase space and trajectory flows provide a way to efficiently reason about phase-space structures and search for global control paths. The local trajectory generator uses the flow information about the phase-space trajectories to produce smoothed trajectories. The trajectory follower tracks the planned reference trajectory, reactively corrects deviations, and resynthesizes the reference trajectory if the dynamics of the system changes significantly.

4.2.2 Intelligent navigation in phase space

The control objective for a stabilization problem is to synthesize a control path, along which the physical system can be brought to the goal state and made to stay there afterwards under control. We are particularly interested in cases in which physical plants to be controlled operate in large regions where nonlinearities of the plants cannot be ignored. When no global stabilization control laws can be found for the desired state with traditional control techniques, composite control paths have to be synthesized. Geometrically, this is the case where initial states of the systems are far away from the desired state. Phase Space Navigator takes advantage of the underlying dynamics of the phase-space flows to plan the trajectory both locally and globally and switches control at carefully planned time instances and places in phase space[1]. This type of control via phase-space path planning requires relatively smaller control authority and achieves the desired control properties otherwise impossible to obtain or difficult to manually synthesize. It is a small, opportunistic dynamical alteration based on the global knowledge of phase-space structures. As soon as the system enters the neighborhood of the desired state, a local linear controller stabilizes the system at the desired location.

4.2.3 Planning control paths with flow pipes

The geometric modeling of a phase space with flow pipes makes the phase-space control planning and navigation feasible. Given a discrete set of possible control actions, the search for a control path from an initial state to a destination is a reachability problem, *i.e.*, the problem of finding a sequence of connected path segments each of which is under a single control action, as schematically illustrated in Figure 4.1. This point-to-point planning can be naturally executed in the flow-pipe representation of phase spaces: the system can travel along one flow pipe for a while, switch to a new control action, jump onto another flow pipe, and eventually arrive at the goal.

To make this approach computationally feasible, the phase portraits of the dynamical system indexed by different control actions are first parsed into a discrete set of trajectory flow pipes. These flow pipes are then aggregated to intersect each other and pasted together to form a graph, the *flow-pipe graph*. The flow-pipe graph is a directed graph where nodes are intersections of flow pipes and edges are segments of flow pipes. The graph may contain cycles. The initial state and the goal state are nodes in the graph. Each edge, a segment of a flow pipe, of the graph is weighed according to traveling time, smoothness, *etc.* The weight can be a single value or a range of values. With this representation, the search for optimal paths is formulated as a search for shortest paths in the directed graph.

Since a flow pipe models a bundle of similar trajectories, it provides room for a small deformation of the reference trajectory within the flow pipe at runtime. This is useful as the synthesized reference trajectory often has to be modified to accommodate uncertainties and noise.

---

[1] Variable-structure control [23] also composes phase spaces along switching surfaces. However, our method differs fundamentally from the variable-structure control in how the phase-space trajectory flows are modeled and utilized in the search for global control paths.

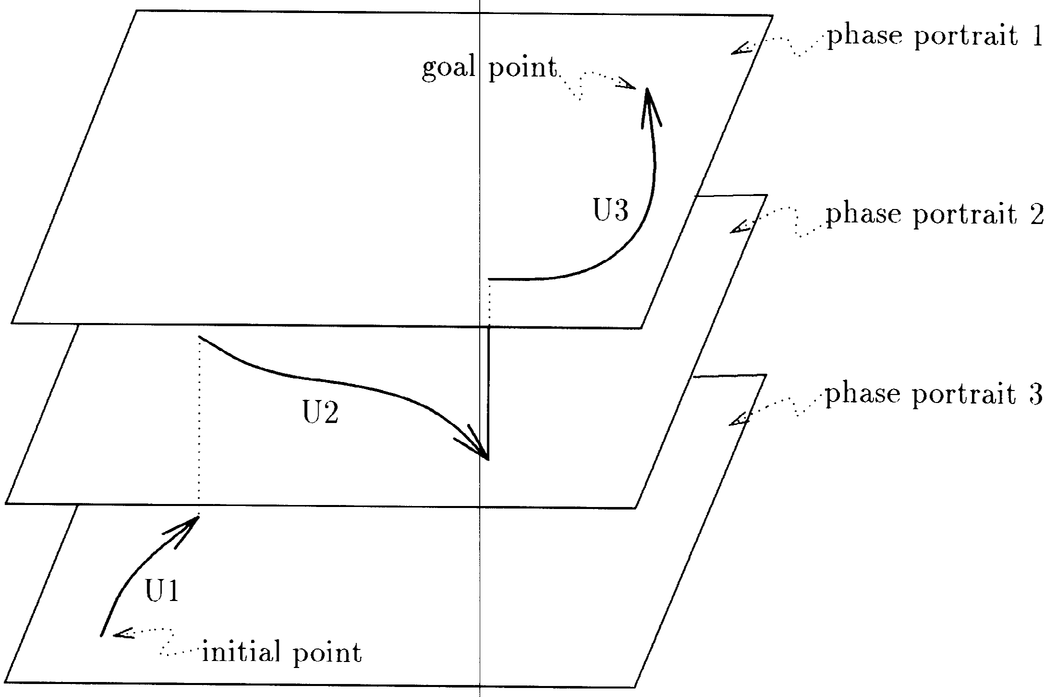

phase portrait 1
goal point
U3
phase portrait 2
U2
phase portrait 3
U1
initial point

Figure 4.1: Search for a control path from an initial point to a goal point in a stack of phase portraits.

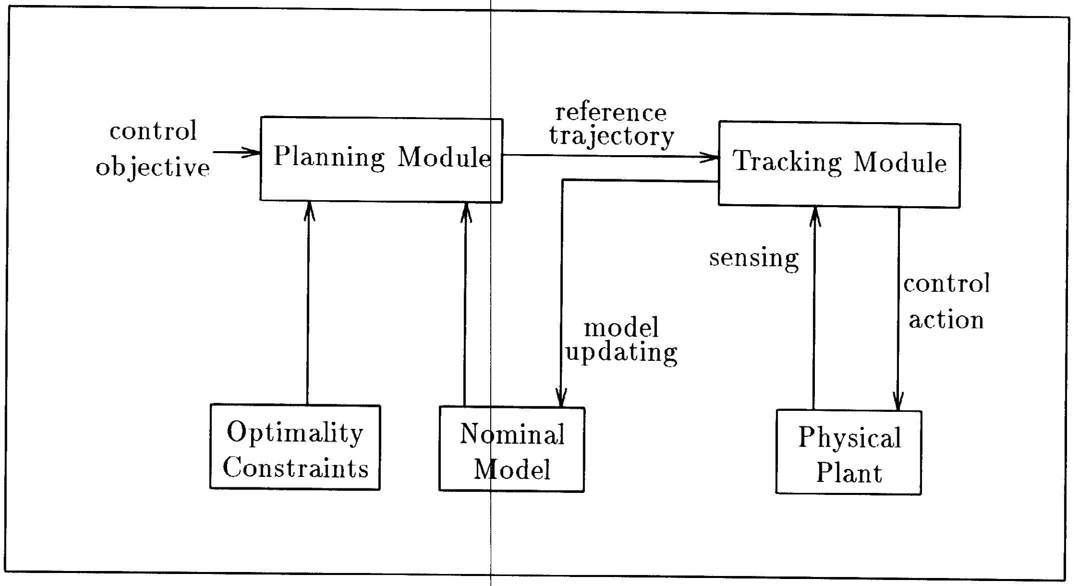

control objective
Planning Module
reference trajectory
Tracking Module
sensing
control action
model updating
Optimality Constraints
Nominal Model
Physical Plant

Figure 4.2: The Phase Space Navigator

4.3 The Phase Space Navigator

We present a general, two-stage architecture for the autonomous synthesis of nonlinear controllers in phase space. The Phase Space Navigator serves as the prototype for the architecture. It has two main modules: the planning module and the tracking module. The planning module synthesizes a desired trajectory, the reference trajectory, in phase space based on the dynamics of the nominal model. The tracking module follows the reference trajectory and reactively corrects local deviations. The parameters of the nominal model are estimated at runtime and the model is updated. When the dynamics changes significantly, the reference trajectory is resynthesized. Figure 4.2 shows the interplay between the two modules in the context of controlling a real physical plant.

4.3.1 Reference trajectory generation

The planning module consists of a global path planner and a local trajectory generator. The global path planner finds coarse global paths from the initial state to the goal state, utilizing the flow pipes of trajectories. The local trajectory generator fits smooth trajectory segments into the global path by slicing out trajectory segments

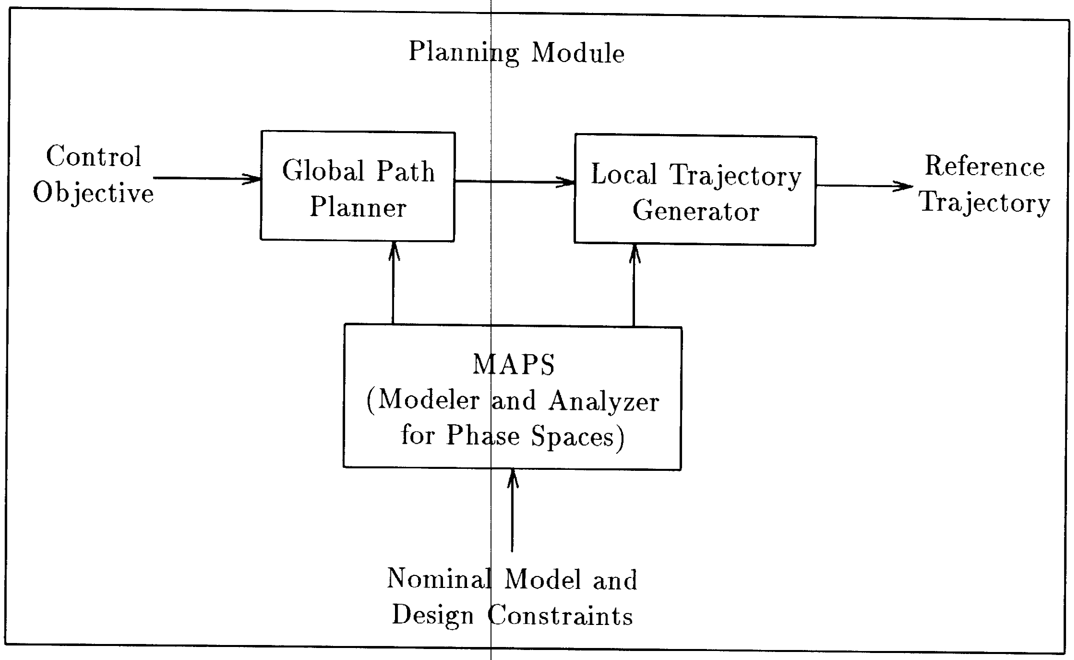

Planning Module
Control Objective
Global Path Planner
Local Trajectory Generator
Reference Trajectory
MAPS (Modeler and Analyzer for Phase Spaces)
Nominal Model and Design Constraints

Figure 4.3: Reference trajectory generation

from flow pipes or through local trajectory deformation. Figure 4.3 illustrates the data flows among components of the planning module.

Global path planner

The global path planner searches for optimal paths from an initial state to a goal state in the phase space, subject to design constraints such as fast convergence and little overshooting. The high-level qualitative description of the phase space guides the search.

To synthesize a global path, MAPS first explores in a set of phase portraits indexed by the selected values of the control signal. It generates a sequence of phase-space descriptions summarized in a geometric vocabulary. Flow pipes are then extracted from the descriptions. The global planner searches for a pipe path — a sequence of flow pipes interconnected at intermediate points — from the initial state to the goal state in this collection of flow pipes.

For stabilization problems, we need to synthesize an attractor at the goal state. When the system enters the neighborhood of the goal state, a linear feedback controller is switched in to stabilize the system at the goal state.

Local trajectory generator

The local trajectory generator produces smoothed trajectories connecting intermediate points. More specifically, trajectory segments are extracted from the flow pipes forming the global path. A trajectory segment is extracted from a flow pipe and is continuously deformed to connect intermediate points in the path. The local trajectory generator also smooths out sharp interconnections of trajectory segments to reduce undesired transients, with local linear controllers or local sliding surface insertion [45]. The noise and certain type of parametric uncertainties of the synthesized reference trajectory are modeled with a *thickened trajectory.* To guarantee the robustness the thickened trajectory segment is restricted within the flow pipe that contains the original reference trajectory segment. However, the method discussed here assumes that the model has no parametric uncertainties or structural uncertainties that change the topological structures of phase space; these kinds of uncertainties include model order uncertainties or unmodeled high-frequency dynamics.

The planning module generates a plan, *i.e.*, the reference trajectory along with the control action for each segment of the reference trajectory.

4.3.2 Reference trajectory tracking

The tracking module follows the planned trajectory and reactively corrects the deviation due to uncertainties in the modeling of dynamics, disturbances, etc. The tracking is usually local to the region of the phase space containing the trajectory segment; see Figure 4.4. The planned reference trajectory provides the global feedforward reference term. The sensor and estimator measure the actual state, whose difference from the reference term is the local feedback correction term.

Trajectory tracking and reactive correction

The Phase Space Navigator tracks a synthesized reference trajectory by sensing the state of the system, comparing the state with the reference trajectory, and computing the current control action. This tracking requires that the full state of the system be observable. If the system stays within the tolerance of the reference trajectory, the controller simply looks up a control action in the planned trajectory. If the system deviates from the desired trajectory, a local linear controller is switched in to force the system back on track. The trajectory correction is local

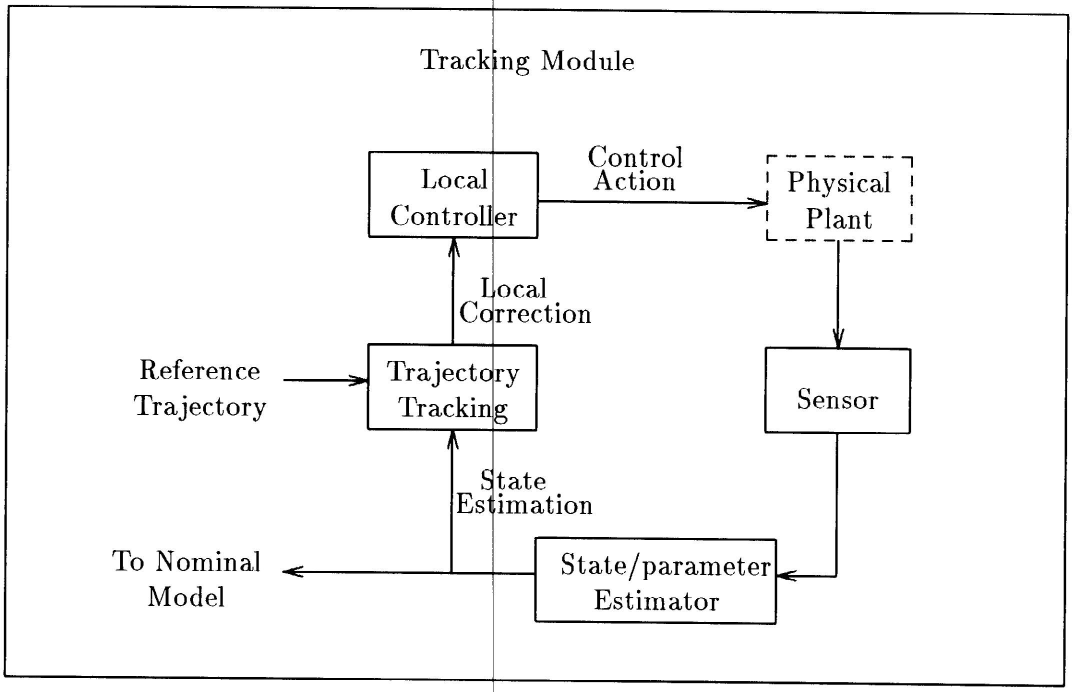

Tracking Module
Local Controller
Control Action
Physical Plant
Local Correction
Reference Trajectory
Trajectory Tracking
Sensor
State Estimation
To Nominal Model
State/parameter Estimator

Figure 4.4: Reference trajectory tracking

and reactive. It is designed to ensure the robustness of the system in the presence of noise in measurements and small changes in system parameters that do not lead to bifurcations.

Model updating and reference trajectory resynthesis

The tracking module estimates the model parameters through the observables and updates the nominal model. When the dynamics of the system changes substantially such that the planned global path is no longer feasible, the Phase Space Navigator calls the planning module to synthesize a new reference trajectory.

4.3.3 The autonomous control synthesis algorithms

We consider a control system $x' = f(x, u)$, where $x \in R^n$ is the state variable and $u \in R^m$ is the control parameter. The initial state of the system is x_s and the goal state for the system to reach is x_g. The set of admissible values for the control parameter u is $\{u_i\}$. The planning algorithm searches for paths from x_s to x_g, subject to certain design constraints, in the phase spaces indexed by control parameter values $\{u_i\}$. The tracking algorithm follows the synthesized reference path.

Algorithm 4.1 *The trajectory planning algorithm:*

1. **Generating phase portraits and parsing them into flow pipes:**

   Generate a collection of phase portraits indexed by $\{u_i\}$. Parse each of the phase portraits into flow pipes.

2. **Constructing flow-pipe graph:**

   Aggregate flow pipes to form flow-pipe graph. Make x_s and x_g to be nodes in the graph. Weigh each edge of the graph with the cost of traversing the edge.

3. **Testing reachability of goal x_g:**

   Cluster the flow-pipe graph into flow-pipe path components. If x_s and x_g are in the same flow-pipe path component, go to Step 6. Otherwise, the goal is unreachable with the given set of parameter values $\{u_i\}$; continue.

4. **Searching for partial paths from x_s to x_g:**

   Compute the reachable set R_{x_s} from x_s and the reverse reachable set R_{x_g} from x_g in the flow-pipe graph. Denote the gap between R_{x_s} and R_{x_g} to be $G = \|R_{x_s} - R_{x_g}\|$.

5. **Tuning control parameter:**

   Search for a value of the control parameter u outside $\{u_i\}$ such that the corresponding phase portrait has flow pipes that, when added to the flow-pipe graph, reduce the value of G. If G is zero, i.e., the gap between R_{x_s} and R_{x_g} is bridged, collect the flow-pipe paths from x_s to x_g and go to Step 7. Otherwise, repeat Step 5 until there are no more parameter values to search, in which case return the collection of partial flow-pipe paths from x_s to x_g.

6. **Finding flow-pipe paths from x_s to x_g:**

   If looking for the shortest paths from x_s to x_g, run a standard shortest path algorithm on the graph. If searching for all paths p from x_s to x_g subject to the constraint $C(p) \leq C$, use the following algorithm[2].

```
(a) let $P_{global-paths} = \emptyset$; $P_{partial-paths} = \{(x_s, x_s)\}$.
(b) for each path $p_i \in P_{partial-paths}$, ending at $x_i$, do
        $P_{p_i} = \{p_i * (x_i, x)|x$ one edge reachable from $x_i$, $x \neq x_g$,
            $C(p_i) + W(x_i, x) \leq C\}$;
        $P'_{p_i} = \{p_i * (x_i, x)|x$ one edge reachable from $x_i$, $x = x_g$,
            $C(p_i) + W(x_i, x) \leq C\}$;
        $P_{partial-paths} = (P_{partial-paths} - p_i) \cup P_{p_i}$;
        $P_{global-paths} = P_{global-paths} \cup P'_{p_i}$.
(c) if $P_{partial-paths} \neq \emptyset$, goto (6b); otherwise, continue.
(d) if $P_{global-paths} = \emptyset$, goto Step 4; otherwise, continue.
(e) order paths in $P_{global-paths}$ according to their costs and return.
```

7. **Generating trajectory segments:**

   Select a representative trajectory segment from each flow pipe forming the flow-pipe path from x_s to x_g and paste them together at intersections of flow pipes.

8. **Smoothing trajectories:**

   Smooth each intersection of trajectory segments through trajectory homotopy deformation. Order the smoothed trajectories with costs and return.

Algorithm 4.2 *The trajectory tracking algorithm:*

---

[2] Note that in the discussion $C(p_i)$ is the cost of a path p_i, $W(x_i, x_j)$ is the weight of the edge (x_i, x_j), and "$*$" is the path composition operator.

1. **Sensing and estimating the physical system:**

   Sense the state x of the physical system, estimate the model parameters, and update the nominal model of the system. If x is in the neighborhood of the goal x_g, go to Step 5. If the nominal model changes substantially, signal and go to Step 4. Otherwise, continue.

2. **Computing control action:**

   Based on the synthesized reference trajectory and the observation from Step 1, compute control tracking term u_{track} and correction term $u_{correct}$. If $u_{correct}$ is unobtainable with a local linear correction, check alternative paths to the goal from the collection of suboptimal paths. If the goal is unreachable, go to Step 4. Otherwise, the control action is $u = u_{track} + u_{correct}$; continue.

3. **Generating control action:**

   Tune the control parameter to u_{track} and generate $u_{correct}$ with a local linear feedback controller. Drive actuators and go to Step 1.

4. **Resynthesizing reference trajectory:**

   Call the planning module to resynthesize a reference trajectory and go to Step 2.

5. **Goal stabilizer:**

   Stabilize the system at x_g with a local linear controller, subject to uncertainties and noise.

Figure 4.5 shows the flow chart of the planning algorithm. The algorithm takes as input the system model, allowable control parameter values, design constraints, and desired control objectives. It outputs a synthesized reference trajectory in the form of a list of tuples: (time, switching point, control parameter value). The planning module has been implemented in Scheme. We will describe how the planning module synthesizes a control law for stabilizing a buckling column. The tracking algorithm described above provides a conceptual framework, whose implementation is an immediate goal of future research.

4.3.4 Discussion of the synthesis algorithms

On-line and off-line synthesis

The planning module generates smoothed reference trajectories in the phase space. The constraints on response time and the availability of computational resources

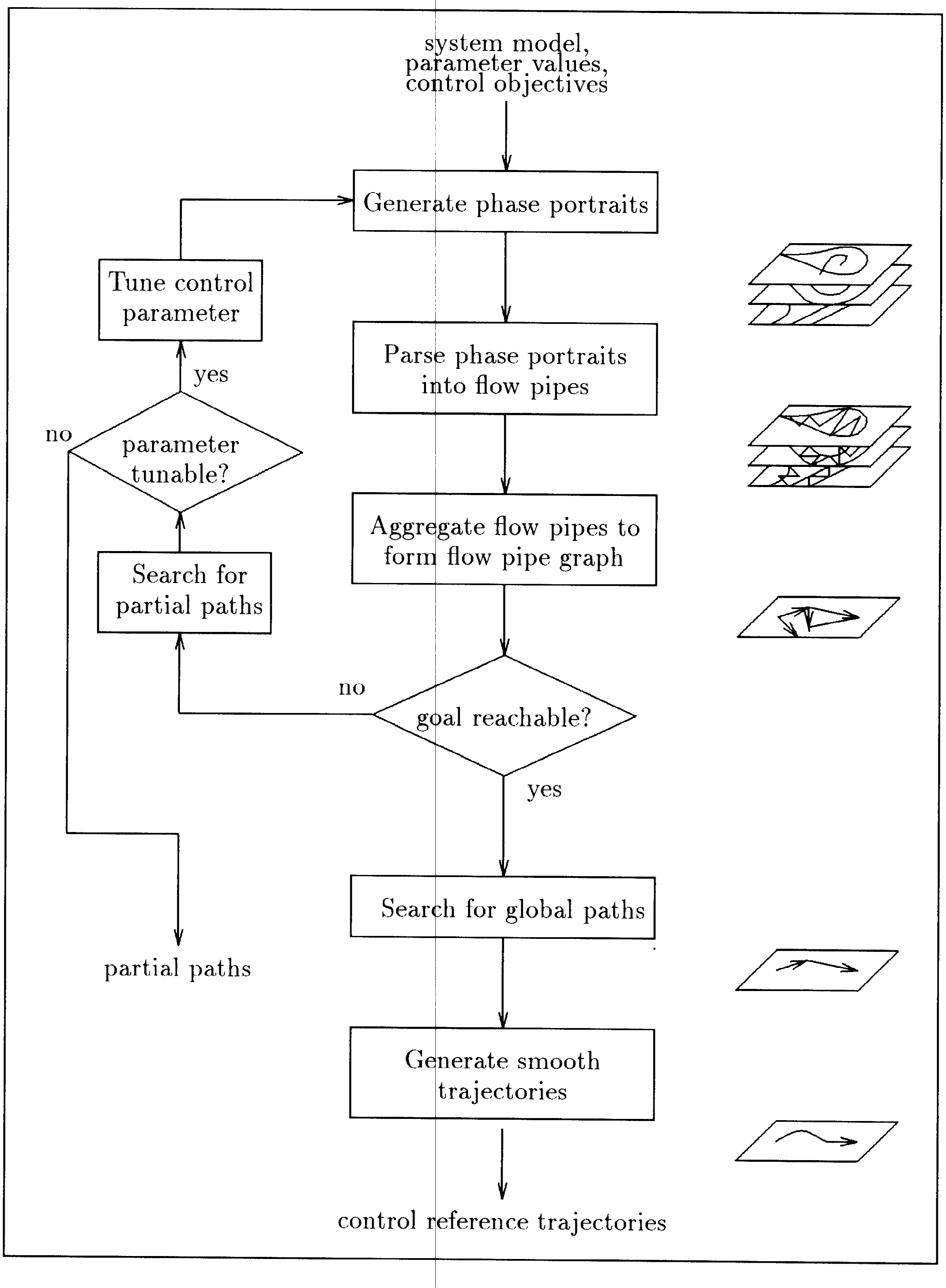

system model,
parameter values,
control objectives
Generate phase portraits
Tune control parameter
Parse phase portraits into flow pipes
yes
no
parameter tunable?
Aggregate flow pipes to form flow pipe graph
Search for partial paths
no
goal reachable?
yes
Search for global paths
partial paths
Generate smooth trajectories
control reference trajectories

Figure 4.5: The flow chart of the trajectory planning algorithm of the Phase Space Navigator.

dictate whether the computation of planning is done on-line or off-line. If the planning is done off-line, the synthesized plan is compiled into a table. At the run-time the controller performs a table lookup for a control action. On-line reference trajectory synthesis and tracking require substantial computational power. The Supercomputer Toolkit [4] provides an ideal platform for experimenting with the above control synthesis algorithm on real-time control applications.

In Step 1 of Algorithm 4.1 for planning, the phase-space structure and flow descriptions can be either computed once for all the control parameter values or generated on demand in the search for optimal paths. The latter case will be more suitable for on-line synthesis.

Generalizations of the point-to-point planning

We present the planning algorithm for a control design problem with one initial state and one goal state. The algorithm also applies to control systems of "one initial state/many goal states", "many initial states/one goal state", or "many initial states/many goal states".

For a "one initial state/one goal state" or "one initial state/many goal states" problem, the Dijkstra's algorithm for the single source shortest paths in a directed graph runs in $O(V^2)$ for a graph with V vertices and E edges [11].

A "many initial states/one goal state" problem can be converted to the "one initial state/many goal states" problem by reversing the directions of all edges; and a "many initial states/many goal states" problem is solved by the Floyd-Warshall algorithm in $O(V^3)$ [11].

Discrete vs. continuous control parameter spaces

A version of the synthesis algorithm is presented for the control parameter initially taking values from a finite discrete set. An example of such finite-valued control systems is the satellite attitude controller, which stabilizes the antenna direction of the satellite subject to disturbances by turning on or off high thrust jets. Other such examples are switching power regulators.

The method also applies to control systems with continuous, multiple control parameter spaces. The program uniformly samples the continuous parameter space at discrete points and then applies the algorithm on these control points. A much better approach, however, would search for "land-mark points" that delimit distinct behaviors and then partition the parameter space into equivalent subspaces. For example, the Bifurcation Interpreter [3] can be employed to search for the bifurcation points defining qualitative changes in dynamics and decompose the parameter space into topologically equivalent subparts.

Suboptimal control paths

The algorithm finds optimal paths in the flow-pipe graph. However, the suboptimal paths can be useful when the optimal paths are no longer judged feasible. They can be stored in the table in addition to the optimal ones. A controller can opportunistically choose among available trajectory paths according to the desired properties at a control switching point.

Comparison with dynamic programming

To synthesize an optimal control path, dynamic programming discretizes a state space and conducts a fine-grain search in the discretization. The cost of the exhaustive search could be prohibitive for large regions or in higher dimensions. In contrast, the Phase Space Navigator searches for control paths in a manageable set of flow pipes. It is possible to use dynamic programming within a flow pipe, once the global path has been established.

4.4 An Example: Stabilizing a Buckling Column

We illustrate the mechanism of the Phase Space Navigator with a control synthesis for stabilizing a buckling steel column. The buckling motion of the column has been extensively studied in nonlinear dynamics from a theoretical point of view and in structural engineering by practicing engineers. The columns are commonly used as strengthening elements in structures; for example, flexible space structures use columns in large operating regions. Study in nonlinear dynamics shows that a slender steel column can exhibit very complicated dynamical patterns under various operating conditions. Therefore, it is important to understand the dynamical behaviors of the columns and to devise ways of preventing them from failure. We will use the Phase Space Navigator to analyze the behaviors of the column and to synthesize a nonlinear controller for stabilizing the column.

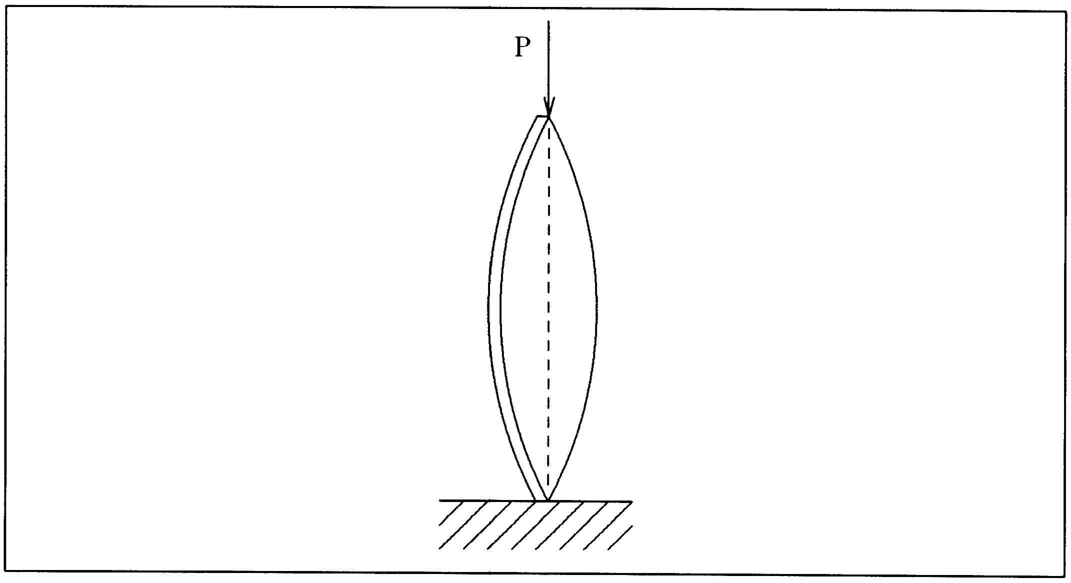


Figure 4.6: Buckling of a thin elastic steel column due to axial end loads.

4.4.1 The column model

A thin elastic column, subject to axial end compressive loads, buckles around the principal axis, as shown in Figure 4.6. The nonlinearity is introduced by the nonlinear geometric stiffness of the column [31]. Stoker [47] gave a simplified model for the column subject to axial compressive force and viscous damping

$$mx'' + cx' + a_1x + a_3x^3 = 0, \tag{4.1}$$

where $a_1 = A + C - 2P/l$ and $a_3 = B + D - P/l^3$. The state x is the characteristic measure of the column deflection from the principal axis and x' is the velocity. The column has mass m and length $2l$. The axial load is P, and the coefficient of viscous damping is c. The bending stiffness is modeled by a primary hard spring with restoring force $Ax + Bx^3$ and a secondary hard spring with restoring force $Cx + Dx^3$. We rewrite equation (4.1) as a system of first-order equations

$$\begin{cases} x_1' = x_2 \\ x_2' = \frac{1}{m}(-a_1x_1 - a_3x_1^3 - cx_2), \end{cases} \tag{4.2}$$

where x_1 represents the deflection and x_2 represents the velocity.

The system (4.2) describes the buckling motion of the column and represents

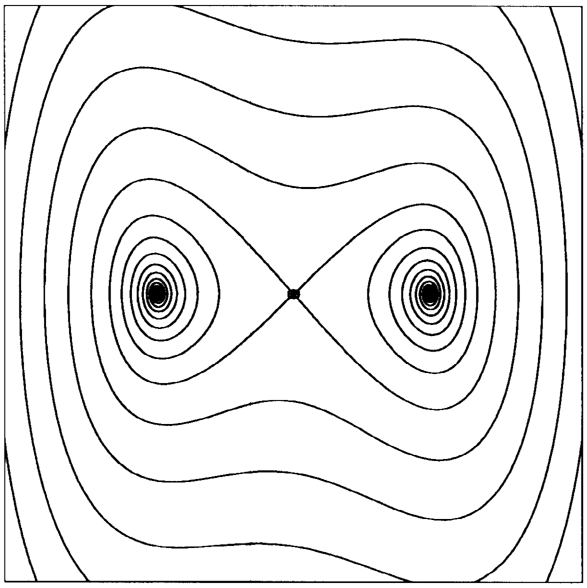


Figure 4.7: The buckling column: the phase space of a buckling column showing the stability boundaries and connecting trajectories. The horizontal axis x is the characteristic measure of the column displacement from its principal axis and the vertical axis x' is the velocity.

only a single mode of vibration. For a long and slender column, vibrations are observed to occur primarily in the first mode [32].

When the axial load P is less than the critical load $P_{critical}$, the column oscillates around the principal axis and returns to the vertical state. It has only one stable state corresponding to the attractor at the origin of the phase space. Under a heavier load, *i.e.*, $P > P_{critical}$, the column buckles to one side or the other. The phase space has a saddle at the origin and two attractors symmetrically arranged about the saddle; see Figure 4.7. The positions of the two buckled states depend on the external load P, the stiffness of the column, and the length l. The larger the load is, the farther away the buckled states are from the principal axis. The column breaks when the buckling exceeds certain limit. As P surpasses $P_{critical}$, the column undergoes a pitchfork bifurcation of equilibria [18]: the attractor at the origin gives birth to a saddle at the origin and two attractors on two sides.

4.4.2 Extracting and representing qualitative phase-space structure of the buckling column

MAPS automatically analyzes the column model in the phase space and extracts and represents the qualitative phase-space structure [54]. For parameter values $a_1/m = -2.0$, $a_3/m = 1.0$, and $c/m = 0.2$ and phase-space region $-3.0 \leq x_1 \leq 3.0$ and $-4.0 \leq x_2 \leq 4.0$, the program reports the following findings and represents them internally in a relational graph:

```
<equilibrium-points:
 equilibrium 1. (attractor at (1.41 0.))
 equilibrium 2. (saddle at (0. 0.))
 equilibrium 3. (attractor at (-1.41 0.))>

<trajectories:
 <boundary-trajectories:
  trajectory 1. (from *infinity* to (0. 0.))
  trajectory 2. (from *infinity* to (0. 0.))>
 <connecting-trajectories:
  trajectory 3. (from (0. 0.) to (-1.41 0.))
  trajectory 4. (from (0. 0.) to (1.41 0.))>>

<stability-regions:
 stability-region 1.
   attractor at *infinity*
   stability-boundary: ()
   connecting-trajectories: ()
 stability-region 2.
   attractor at (1.41 0.)
   stability-boundary: (trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 4)
 stability-region 3.
   attractor at (-1.41 0.)
   stability-boundary: (trajectory 2. trajectory 1.)
   connecting-trajectories: (trajectory 3)>
```

The program finds two attractors at $(1.41, 0.0)$ and $(-1.41, 0.0)$ and a saddle at the origin. It generates a high-level description of the phase-space geometry: two banded stability regions associated with the two attractors, separated by the stable trajectories of the saddle at the origin. Figure 4.7 shows stability boundaries

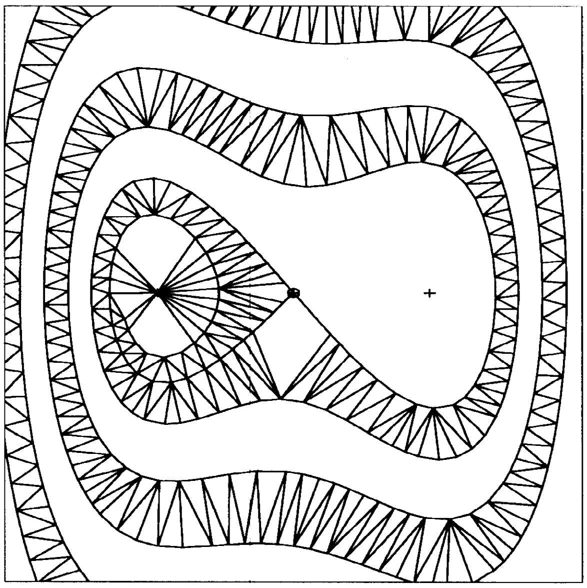


Figure 4.8: The buckling column: the flow pipe leading to the attractor on the left.

and connecting trajectories of the two stability regions. Based on the geometric phase-space representation, the phase space is further modeled with two flow pipes formed by aggregating geometric pieces, using the flow-pipe construction algorithm. The pipe boundaries consist of the separatrices of the two stability regions that approach the saddle and of the trajectories that connect equilibria, as described by the program in the following:

```
<flow-pipes:
 flow-pipe 1. (from *infinity* to (-1.41 0.))
   boundary: (trajectory 2. trajectory 1. trajectory 3.)
 flow-pipe 2. (from *infinity* to (1.41 0.))
   boundary: (trajectory 2. trajectory 1. trajectory 4.)>
```

Figure 4.8 shows the flow pipe that ends at the left-hand attractor.

4.4.3 Synthesizing control laws for stabilizing the column

We want to stabilize the column at its vertical state to support the axial end loads and to prevent it from breaking. Under sufficiently heavy load, the buckling motion of the column leads the column to one of the buckled states and, when the states are far away from the principal axis, induces the failure of the column.

We shall focus on global navigation in phase space that synthesizes global reference trajectories leading to the desired goal state. Since the design of local linear controllers is relatively well understood, we shall not discuss the trade-offs among different designs for linear controllers and shall choose, for the purpose of demonstration, a simple linear feedback design. Local controllable regions of such linear controllers are quantified, given available control strength, and are used in constraining the design of global control paths.

The goal state of the control is the unbuckled state—the saddle at the origin of the phase space that does not have a stability region. We want to synthesize a non-zero stability region for the goal and maximize the region. The controlled column is of the form

$$\begin{cases} x_1' = x_2 \\ x_2' = \frac{1}{m}(-a_1x_1 - a_3x_1^3 - cx_2) + u, \end{cases} \tag{4.3}$$

where u is the control. In the model, mu has the same dimension as the force.

The Phase Space Navigator automatically synthesizes a global trajectory from an initial state x_s to the goal state x_g. We consider two cases for the initial state ${x_s}$[3].

Control design I: stabilizing the buckling motion

The column is initially buckling with sufficient velocity. The initial state x_s is far away from the saddle x_g in phase space in this case. The control parameter u takes its values from a small range around zero.

The Phase Space Navigator constructs a flow-pipe graph from phase spaces of $u = 0$ and $u =$`*local-control*`, as shown in Figure 4.9. The graph shows the case when the initial state x_s is in the flow pipe ending at the left attractor as shown in Figure 4.8. Denote by U_{ctl} the local controllable region at the goal x_g. The two flow pipes of $u = 0$ intersect U_{ctl} at u_1 and u_2 respectively. The edges $\overline{u_1x_g}$

---

[3] We use this example for the purpose of illustrating the Phase Space Navigator. The system (4.3) is actually feedback linearizable. A feedback linearization would cancel the nonlinearity of the vector field in x_2 direction. In contrast, our design restrains the control to be less than 10% of the vector-field strength in the first case and less than half of the vector-field strength in the second case.

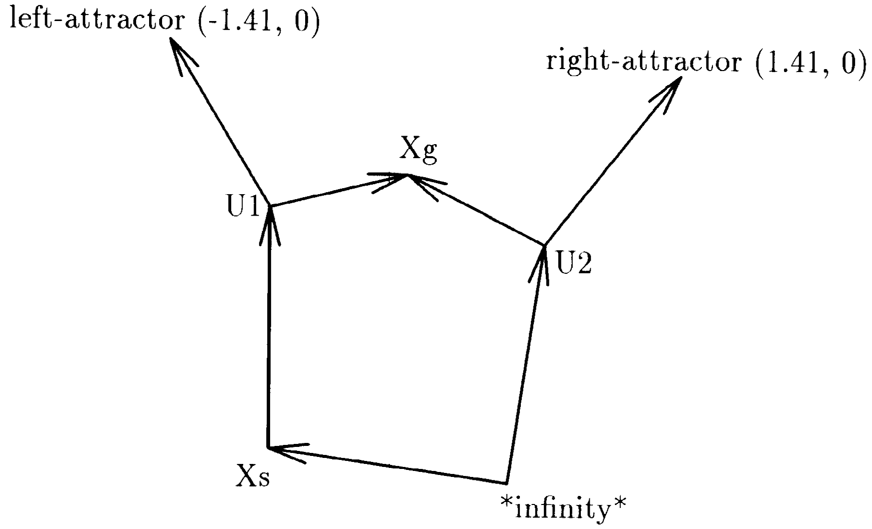


Figure 4.9: The flow-pipe graph for the buckling column.

and $\overline{u_2x_g}$ represent flows within U_{ctl} produced by the local controller. The paths ($*infinity* \rightarrow x_s \rightarrow u_1 \rightarrow$ *left-attractor*) and ($*infinity* \rightarrow u_2 \rightarrow$ *right-attractor*) represent the two flow pipes of the phase space of $u = 0$.

The Phase Space Navigator finds a simple path from x_s to x_g in the graph, consisting of edges $\overline{x_su_1}$ and $\overline{u_1x_g}$. Then the program synthesizes an individual trajectory connecting x_s and x_g from the flow-pipe path ($\overline{x_su_1}$, $\overline{u_1x_g}$). To construct this desired trajectory, the program deforms an uncontrolled trajectory, emanating from x_s, of the flow pipe represented by the edge $\overline{x_su_1}$ so that the deformed trajectory enters U_{ctl}. The region U_{ctl} is first projected backwards through the flow pipes to form a goal projection. Figure 4.10 shows the goal projection from U_{ctl}— two thin pipes illustrated in thick solid lines. Then the uncontrolled trajectory is deformed towards the goal projection. The programmed deformation is designed to push the trajectory towards the nearest goal projection in the controllable direction x_2. Since the control is more effective when the direction of the vector field is relatively orthogonal to the controllable direction x_2, switching points are inserted to turn off the controller when the angle between the two directions are below some threshold.

Consider for example the case when the column is initially at the state $(-1, -3)$. The control design is specified as:

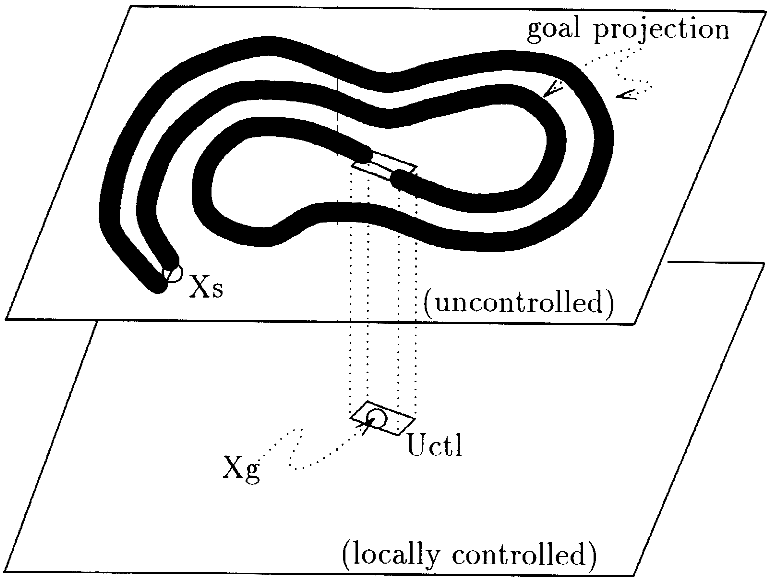


Figure 4.10: The goal projection and the deformation of the trajectory.

control_type:	point_to_point
goal_state:	(0.0, 0.0)
initial_state:	(-1.0, -3.0)
range_of_control:	$u \in$ [-0.2,0.2]

The Phase Space Navigator designed a control law that brings the column back to the unbent state. The control law is represented as a list of tuples, each of which specifies the time, state, and control value for each switching of control:

```
((time 0.) (switching-state #(-1 -3)) (control .2))
((time .284) (switching-state #(-1.82 -2.71)) (control 0.))
((time 1.06) (switching-state #(-1.86 2.49)) (control -.2))
((time 2.71) (switching-state #(1.35 1.82)) (control 0.))
((time 6.76) (switching-state #(-.0023 -.0692)) (control *local-control*))
```

Figure 4.11(a) shows the synthesized reference trajectory originating at $(-1, -3)$. The circles indicate places where the control of deformation switches. Each segment of the reference trajectory delimited by the switching points is under interval-constant control, as specified by $U1$, $U2$, $U3$, $U4$, or a local linear control law as the trajectory is in the vicinity of the goal. The global portion of the reference trajectory is pushed towards the goal projection with small deformation that is less than 10% of the vector field strength at any state or 0.2 in the normalized unit, whichever is smaller. The position x and velocity $v = x'$ of the controlled column are plotted against the time t in Figure 4.11(b) and Figure 4.12(c), respectively. The control u is shown in Figure 4.12(d).

When the goal is not reachable with the current deformation, the synthesis algorithm tries again with increased control strength if possible. The synthesized trajectories could be further optimized with variational techniques on the collection of trajectories within the flow-pipe segments [6].

Control design II: restoring from the buckled state

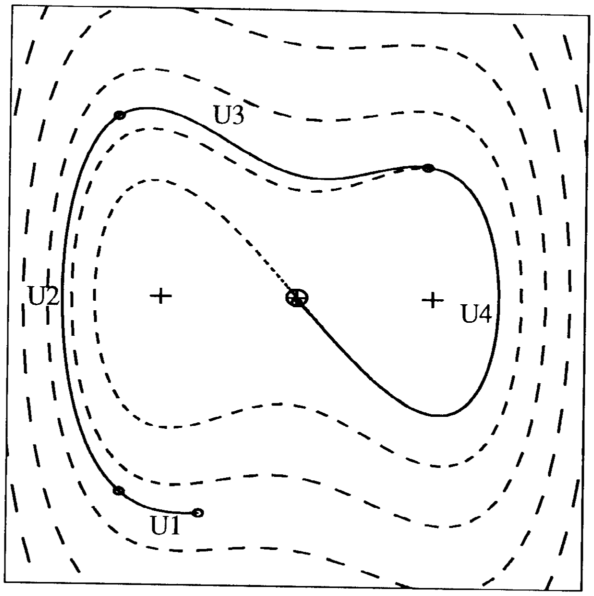
The column is initially near the buckled state $(-1.41, 0.0)$. The control objective here is to pull the column out of the buckled state and to bring it close to the unbuckled state. Since the flow pipe containing the initial state does not intersect U_{ctl} in the down stream, a different control strategy must be employed. The program uses the following strategy to synthesize a control path.

Assume the initial state of the column is $(-1.5, 0.0)$. The control objective is:

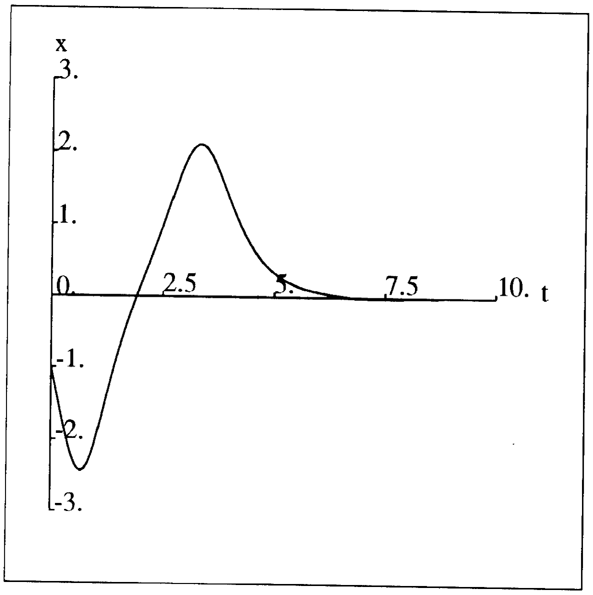
```
control_type:      point_to_point
goal_state:        (0.0, 0.0)
initial_state:     (-1.5, 0.0)
range_of_control:  u ∈ [-1.0, 1.0]
```

The uncontrolled trajectory from the initial state would spirally approach the attractor in the clockwise fashion. The control is exerted in such a way as to swing the trajectory away from the buckled state to approach the goal projection of the saddle. When the trajectory intersects the goal projection, the control is switched off so that the system slides along the uncontrolled trajectory. As soon as the system enters U_{ctl}, the linear controller is switched in. The control strength is less than half of the vector field strength or 1.0 in the normalized unit, whichever is smaller. The synthesized control law is specified as:

```
((time 0.) (switching-state #(-1.5 0)) (control -.187))
((time .001) (switching-state #(-1.5 .000187)) (control .187))
((time 1.65) (switching-state #(-1.24 -.0011)) (control -.289))
((time 3.19) (switching-state #(-1.66 .0153)) (control .638))
((time 5.92) (switching-state #(-.527 -.00186)) (control -.454))
((time 7.74) (switching-state #(-2.02 .0473)) (control 1.))
```

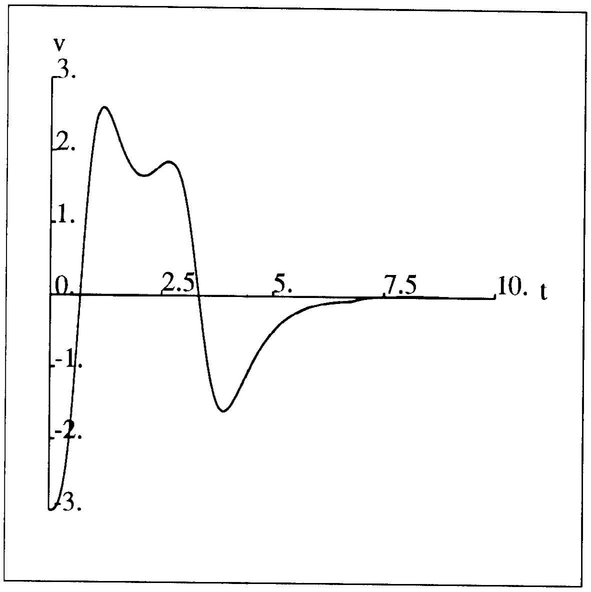


(a)

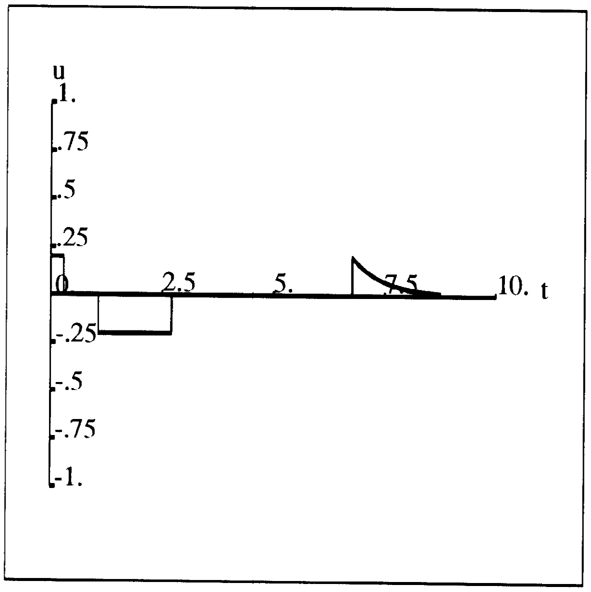


(b)

Figure 4.11: The synthesized control law that stabilizes the buckling column: (a) the reference trajectory that leads to the unbuckled state corresponding to the saddle at the origin. The column is initially buckling with sufficient velocity; (b) the position x of the column plotted against time t. (to be continued)



(c)



(d)

Figure 4.12: (cont'd) The synthesized control law that stabilizes the buckling column: the velocity $v(= x')$ of the column and the control signal u for stabilizing the column are plotted against time t in (c) and (d), respectively.

```
((time 8.08) (switching-state #(-1.75 1.47)) (control 0.))
((time 11.1) (switching-state #(-.049 .00163)) (control *local-control*))
```

The corresponding reference trajectory is shown in Figure 4.13(a). The control parameter changes at places marked by circles. The reference trajectory consists of eight trajectory segments labeled by interval-constant control $U1$, $U2$, ..., $U7$, and a local control law. The first segment ($U1$) and the last segment (local control) are very short in length and thus are invisible in the figure. The position x and velocity $v = x'$ of the controlled column are plotted against the time t in Figure 4.13(b) and Figure 4.14(c), respectively. The control u is shown in Figure 4.14(d).

Control design III: local control near the goal

A simple local linear controller of the form $u_{local-linear} = -k_1 x_1 - k_2 x_2$ is used to stabilize the system around the goal (see [55] for details). Given the maximum control strength 0.2 for the linear controller, we choose an over-conservative region for U_{ctl} determined by

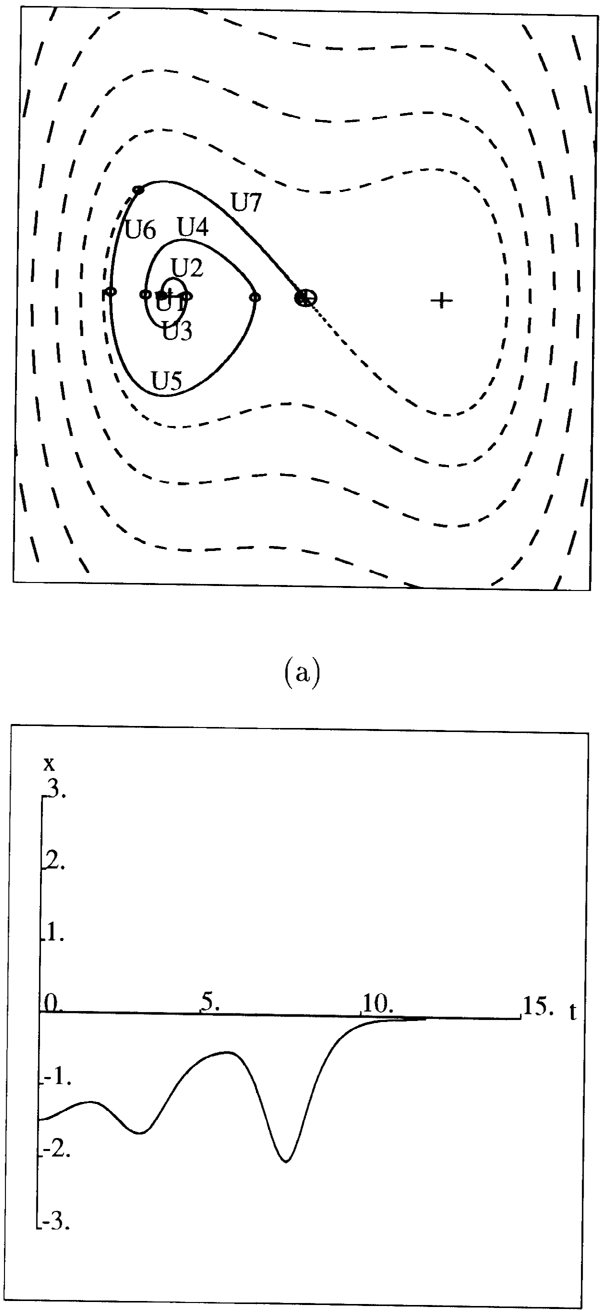
$$U_{ctl} = \{(x_1, x_2) \mid |0.835x_1 + 0.550x_2| \leq 0.04, \ |0.797x_1 - 0.605x_2| \leq 0.04\}. \quad (4.4)$$

This U_{ctl} is used in both of the above cases.

4.4.4 The phase-space modeling makes the global navigation possible

The qualitative description of the phase-space structure and the geometric modeling of the trajectory flows provide a "map" for navigating system trajectory to the goal in phase space.

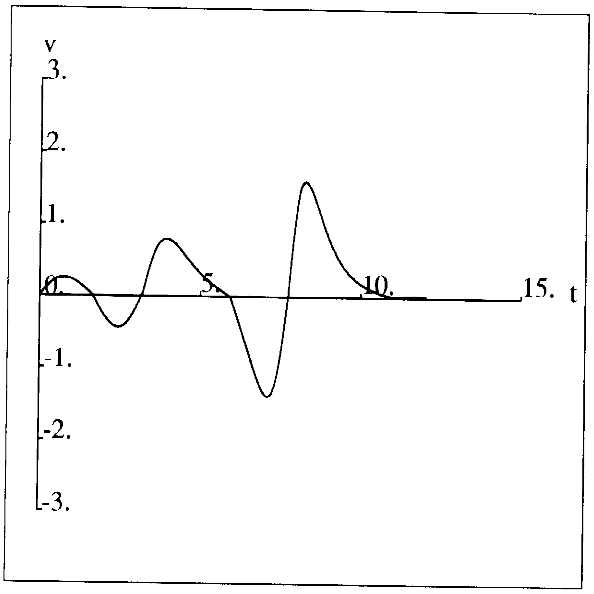
The directed graph constructed from the flow pipes is used in searching for global paths and in determining whether the goal is reachable. The flow pipes also make it possible to characterize the more microscopic spatial and temporal relations between the current state and the goal state. For example, the collection of simplices near the goal forms a neighborhood of the goal that determines whether or not the trajectory has missed the goal in the flow pipe. The deformation of global path segments is constrained by reasoning about the spatial relation with flow pipes.



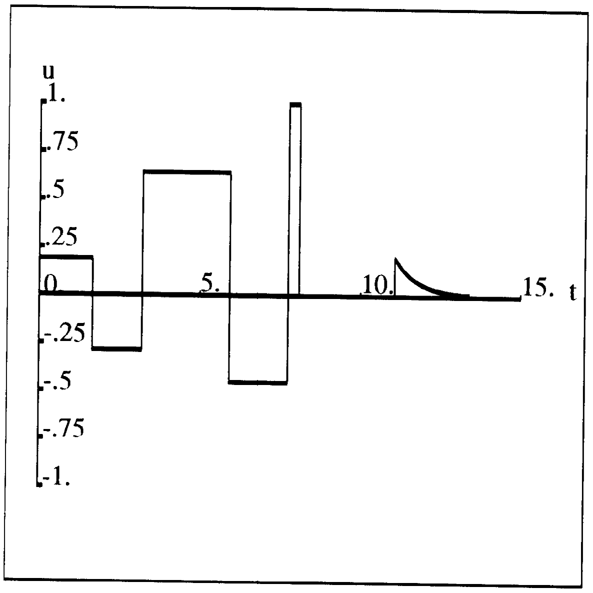
(a)

(b)

Figure 4.13: The synthesized control law for restoring the column from the buckled state: (a) the reference trajectory that swings the column out of the buckled state; (b) the position x of the column plotted against time t. (to be continued)



(c)



(d)

Figure 4.14: (cont'd) The synthesized control law for restoring the column from the buckled state: the velocity $v(= x')$ of the column and the control signal u for controlling the column are plotted against time t in (b) and (c), respectively.

4.4.5 Other control problems

The stabilization for a buckling column is closely related to the pole-balancing problem, where the control objective is to balance a pole at the vertically upward state. In the phase space of the pole model, the saddle is the upward state of the pole and the two attractors on two sides of the saddle correspond to the same downward state of the pole.

The control objective for pole-balancing is to bring the pole to the upward state from any other positions, even when the pole is initially hanging downward. The problem has been studied as a standard testbed for many control designs, most of which focus primarily on linear feedback control in linear regimes. With our phase-space navigation strategy, a globally stable reference trajectory can be synthesized, similar to the one for the buckling column discussed earlier.

The optimal design of such trajectories has many practical implications. For example, the efficiency of cargo handling work at shipyards depends largely on the operation of overhead cranes [44]. Figure 4.15 illustrates such a crane driven by a trolley drive motor and a hoist motor. The bottleneck of the ship unloading operation is the transfer of cargo from a ship to waiting trucks; therefore, minimizing this transfer time brings about a large cost saving. The planar motion of the load hanging at the moving trolley can be modeled as a swinging pendulum with a moving point of attachment and a hoisting cable. The trajectories of such overhead cranes could be optimally designed with the Phase Space Navigator. This would be an important step towards the full automation of cargo handling work without a crane operator.

4.5 Summary of the Chapter

We have developed an autonomous control synthesis method, the Phase Space Navigator, for synthesizing controllers for nonlinear systems in the phase spaces. We have discussed primarily finite-valued control systems as examples to illustrate the method and have noted that the method also applies to control systems with continuous parameter spaces.

The Phase Space Navigator synthesizes global reference trajectories using knowledge of phase-space structures provided by the modeling and analysis program. A phase space is modeled with flow pipes that are collections of trajectories having the same qualitative behaviors. The flow pipes provide a way to efficiently

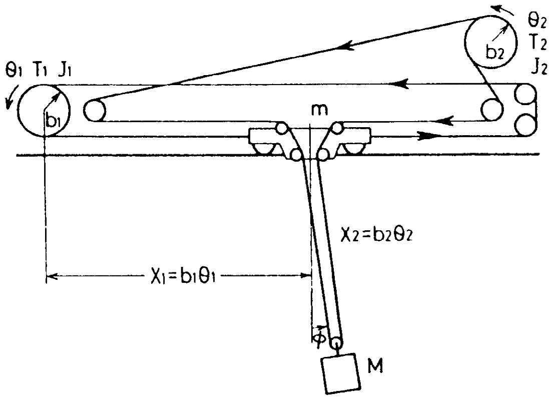


Figure 4.15: The mechanical model for an overhead crane unloading cargo from ships, reproduced from [Sakawa&Shindo, 1982]. The crane is equipped with a trolley drive motor and a hoist motor. The planar motion of the load is modeled as a swinging pendulum hanging at the moving trolley.

search for and reason about global structures of phase spaces. The global nonlinear control synthesis becomes a graph search problem with this representation of the phase space. We have shown how global reference trajectories can be automatically designed with the example of stabilizing a buckling column. With the novel idea of grouping an infinite number of trajectories into a manageable collection of flow pipes, the difficult task of synthesizing a nonlinear controller is formulated as a computational problem that requires a combination of substantial numerical, symbolic, and combinatorial computations and spatial reasoning techniques.

The leverage of the automatic synthesis method comes from applications whose operating regions are grossly nonlinear and on which high-performance global controllers are impossible to synthesize with traditional techniques. In the next chapter, we will demonstrate the application of the Phase Space Navigator to the design of a maglev controller. Other potential engineering applications include large flexible space structures, robot manipulator planning, satellite attitude control, and switching power regulators.

Chapter 5

An Application: Design of a Maglev Controller

This chapter studies a magnetic levitation system and applies the machineries of the Control Engineer's Workbench developed in Chapters 3 and 4 to the design of a stabilizing controller for the system.

The Workbench has helped synthesize a global, nonlinear controller for the nominal model of a German maglev system[1]. We describe the systematic state-space design method for determining the global switching points of the controller. The synthesized control system can stabilize the maglev vehicle with large initial displacements from an equilibrium. The simulation shows that our nonlinear controller possesses an operating region more than twenty times larger than that of the classical linear feedback design for the same system.

5.1 Introduction

Magnetically levitated trains provide a high speed, very low friction alternative to conventional trains with steel wheels on steel rails. Several experimental maglev systems in Germany and Japan have demonstrated that this mode of transportation can profitably compete with air travel. More importantly, maglev transportation can ease traffic congestion and save energy [14, 28, 50].

---

[1] The maglev stabilization problem was suggested to the author by Professor Richard Thornton of Electrical Engineering at MIT, who is leading the MIT Maglev Consortium. The result of the work described here appears in a paper [56] to be presented at the 31st IEEE Conference on Decision and Control, December 1992.

Maglev transportation uses magnetic levitation and electromagnetic propulsion to provide contactless vehicle movement. There are two basic types of magnetic levitation: electromagnetic suspension (EMS) and electrodynamic suspension (EDS). In EMS, the guideway attracts the electromagnets of the vehicle that wraps around the guideway. The attracting force suspends the vehicle about one centimeter above the guideway. In contrast, the EDS system uses repulsive force, induced by the magnets on the vehicle, to lift the vehicle.

5.2 The Maglev Model

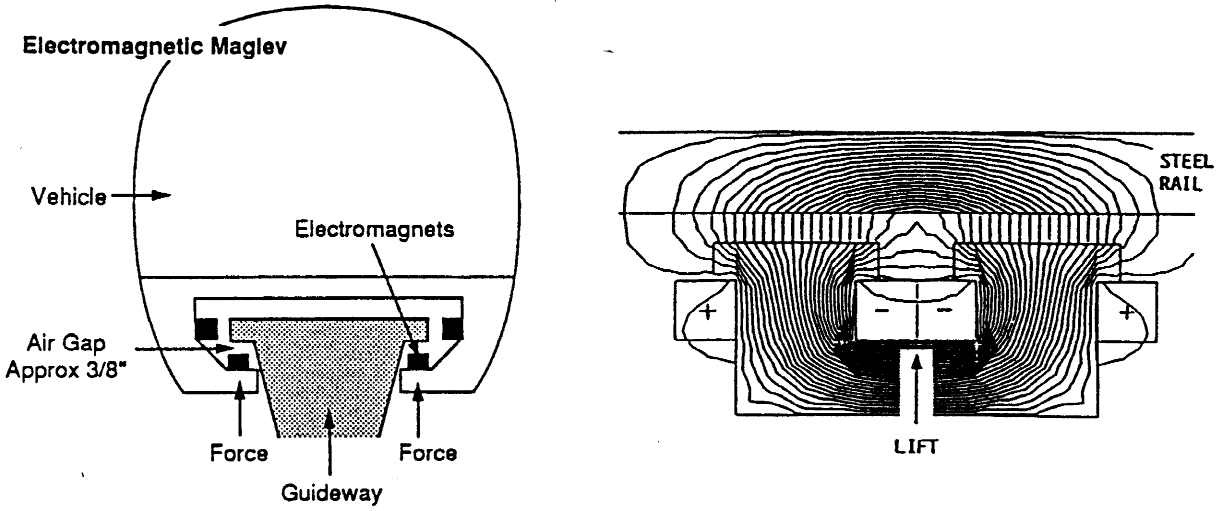
An attractive system such as the EMS system is inherently unstable. We consider the control design for stabilizing an EMS-mode train traveling on a guideway—a simplified model for the German Transrapid experimental system. The Transrapid system is schematically shown in Figure 5.1. It uses attractive magnetic forces to counterbalance gravitational forces.

The state equations for the magnetically levitated vehicle and the guideway are described by

$$\begin{cases} \frac{dx}{dt} = \frac{z(V_i - Rx)}{L_0 z_0} + \frac{xy}{z} \\ \frac{dy}{dt} = g - \frac{L_0 z_0 x^2}{2mz^2} \\ \frac{dz}{dt} = y \end{cases} \tag{5.1}$$

where the state variables x, y, and z represent coil current in the magnet, vertical velocity of the vehicle, and vertical gap between the guideway and the vehicle, respectively. The control parameter is the coil input voltage V_i. The other parameters are the mass of the vehicle m, the coil resistance R, the coil inductance L_0 and the vertical gap z_0 at the equilibrium, and the gravitational acceleration g. Details of the derivation of the model are discussed in [49]. The nonlinearities of the system come from the nonlinear inductance due to the geometry of the magnet and the inverse square magnetic force law.

The system has one equilibrium state at which the magnetic force exactly counterbalances the force due to gravity and the vehicle has no vertical velocity and acceleration. However, the equilibrium is a saddle node which is not stable. The control objective, therefore, is to stabilize the vehicle traveling down the guideway and maintain a constant distance between the vehicle and the guideway despite any roughness in the guideway. The available control input is the coil input voltage V_i in the model (5.1). We further assume that V_i is produced by a buck converter


Electromagnetic Maglev
Vehicle
Electromagnets
Air Gap
Approx 3/8"
Force
Force
Guideway
STEEL
RAIL
+
−
−
+
LIFT

(a) (b)

Figure 5.1: EMS maglev system for high-speed ground transportation, representing a simplified drawing of the German Transrapid design. (a) Electromagnetic suspension (from [MTAC Report 1989]); (b) Detail of a suspension magnet, superimposed on the field distribution (from [Eastham 1989]).

capable of delivering any voltage from 0 to 300 volts.

A linear control design for the maglev system described in [49] uses the pole-placement method. The system is first linearized around the equilibrium. The linearized system has unstable poles, i.e., the poles in the right-half of s-plane. A linear feedback is introduced to move the poles to the desired locations in the left-half of the s-plane. Such a control design can bring the system back to the equilibrium with an initial displacement of up to 0.2mm from the equilibrium. The linear controller saturates at the beginning for larger initial displacements. This is because the linearized model no longer approximates the original system well in regions far away from the equilibrium. A global, nonlinear control law such as a bang-bang control that respects the nonlinearity of the system must therefore precede the linear feedback control. However, the real challenge for the nonlinear design is to determine the global control law specifying, for instance, the switching points.

5.3 State-Space Control Trajectory Design

We will describe a nonlinear control design—a switching-mode control—in state space for the maglev system with large initial displacements from the equilibrium. We will show that this controller can be automatically designed with the Workbench comprising MAPS and Phase Space Navigator developed earlier. The nonlinear controller brings the system to the vicinity of the equilibrium and then switches to the linear controller.

For the purpose of demonstration, we assume that the vehicle is displaced from the equilibrium in the direction further away from the guideway. We will concentrate on the global design of the control reference trajectories and assume that a linear feedback design is available as soon as the system enters the capture region of the linear controller.

5.3.1 Modeling state-space geometry

The global control law is designed by analyzing and modeling the state-space geometry of the system. The Workbench explores the state space of the system and characterizes the state space with stability regions and trajectory flow pipes. It composes the state spaces for different control parameter values and uses flow pipes to synthesize a composite state space.

The state variables x, y, and z in the model are scaled by 1, 10^3, and 2×10^4, respectively. The parameters of model are assumed to be: $L_0 = 0.1h$, $z_0 = 0.01m$, $R = 1\Omega$, $m = 10000kg$, and $g = 9.8m/sec^2$, typical of a large vehicle lift magnet. Assume the power supply delivers 140 volts, *i.e.*, $V_i = 140$, at the equilibrium. The Workbench explores the state space in a region bounded by the box $\{(x, y, z) | x \in [0, 400], y \in [-300, 350], z \in [0, 600]\}$ and finds the following equilibrium point:

```
saddle:       #(140. 0. 200.)
eigenvalues:  -17.004+22.963i
              -17.004-22.963i
              24.007
eigenvectors: #(.23604 .97174 0)
              #(.51331 -.55588 .65384)
              #(.30157 .73255 .61027)
```

With the information about the stable eigenvectors of the saddle, the Workbench computes the stable manifold of the saddle, a two-dimensional surface. The Workbench generates a set of trajectories evenly populating the stable manifold to approximate the surface. The trajectories are obtained by backward integrations from initial points in a small neighborhood of the saddle. This neighborhood lies within the plane spanned by the stable eigenvectors of the saddle.

Figure 5.2 shows the trajectories on the stable and unstable manifolds of the saddle in the yz-projection of the state space. The stable manifold is two-dimensional and the unstable one is one-dimensional. The stable manifold separates the state space into two halves: trajectories in the upper-half approach $z \to \infty$ along one of the unstable trajectories, corresponding to the case in which the vehicle falls off the rail; and trajectories in the lower-half approach $z = 0$ plane along the other unstable trajectory, corresponding to the case in which the train collides with the rail.

5.3.2 Synthesizing a global stabilization law

For an initial displacement above or below the equilibrium, the uncontrolled system will follow either a trajectory traveling upwards with increasing z and leaving the bounding box or a trajectory traveling downwards and hitting the $z = 0$ plane. To stabilize the system at the equilibrium, it is necessary to synthesize a new vector field on both sides of the stable manifold so that trajectories travel towards the

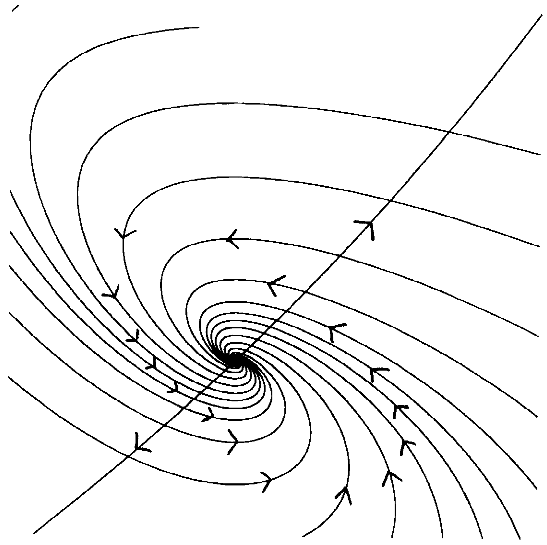


Figure 5.2: Stable and unstable manifolds of the saddle for $V_i = 140$ (yz-projection).

stable manifold of the saddle in the new vector field. We consider only the top-half here.

The Workbench first explores the state space of the model for different values of V_i and concludes that the larger the V_i is, the further away the stable manifold is from the $z = 0$ plane. For $V_i = v > 140$, the region sandwiched by the stable manifold of $V_i = 140$ and that of $V_i = v$ has the desired property—the vector field of $V_i = v$ in this region is pointed towards the stable manifold of $V_i = 140$. When $v = 300$, the region is maximized.

Similarly, the Workbench finds that the model with $V_i = 300$ has a saddle node at $(300., 0., 428.57)$. The stable manifold of the saddle has a similar structure as that of $V_i = 140$ case, but with larger z coordinate. The yz-projection of the stable manifold and unstable trajectories for $V_i = 300$ is shown in Figure 5.3. Appendix A contains the information about the saddle for $V_i = 140$ and $V_i = 300$.

Figure 5.4 shows the yz-projection of the sandwiched region discussed above. The region is bounded by three pieces of triangulated surfaces, the top boundary shown in Figure 5.5(a), the side one in Figure 5.5(b), and the bottom one in Figure 5.5(c), respectively. The top boundary represents the stable manifold of the saddle for $V_i = 300$ and the bottom one for $V_i = 140$. The trajectories of

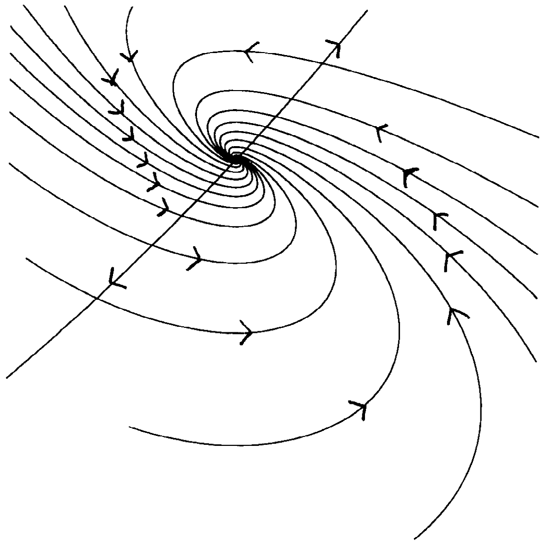


Figure 5.3: Stable and unstable manifolds of the saddle for $V_i = 300$ (yz-projection).

$V_i = 300$ flow into the region from the side boundary in Figure 5.5(b) and leave the region at the bottom boundary in Figure 5.5(c). There is no flow across the top boundary in Figure 5.5(a).

The Workbench determines that the above region consists of two trajectory flows: the one for $V_i = 300$ that flows into the region on the side boundary and pierces through the bottom boundary, and the one for $V_i = 140$ on the bottom boundary that approaches the desired equilibrium in the limit. With the flow-pipe characterization of the state-space trajectory flows, the Workbench searches for control trajectories in this set of flow pipes and finds a sequence of flow pipes that lead to the desired goal: the composite of the trajectory flow for $V_i = 300$ and the flow for $V_i = 140$, glued together at the stable manifold of $V_i = 140$ represented by the bottom boundary of Figure 5.5(c). As a result, all the trajectories of $V_i = 300$ within the region can be brought to the equilibrium by switching to $V_i = 140$ as soon as the trajectories hit the bottom boundary. We call this region the controllable region for the system, as shown in Figure 5.4.

Since the bottom boundary is an approximation to the true stable manifold, the trajectories of $V_i = 140$ on the boundary can only get close to the desired equilibrium. The closeness depends on the quality of the manifold approximation.

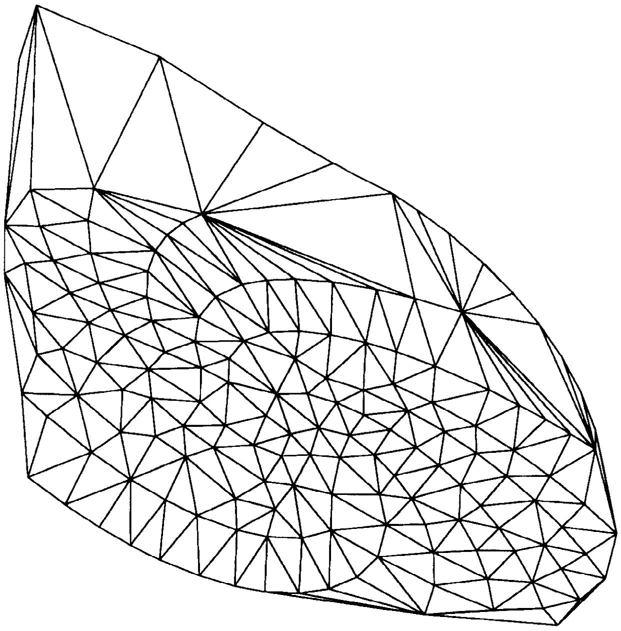


Figure 5.4: The sandwiched region in yz-projection.

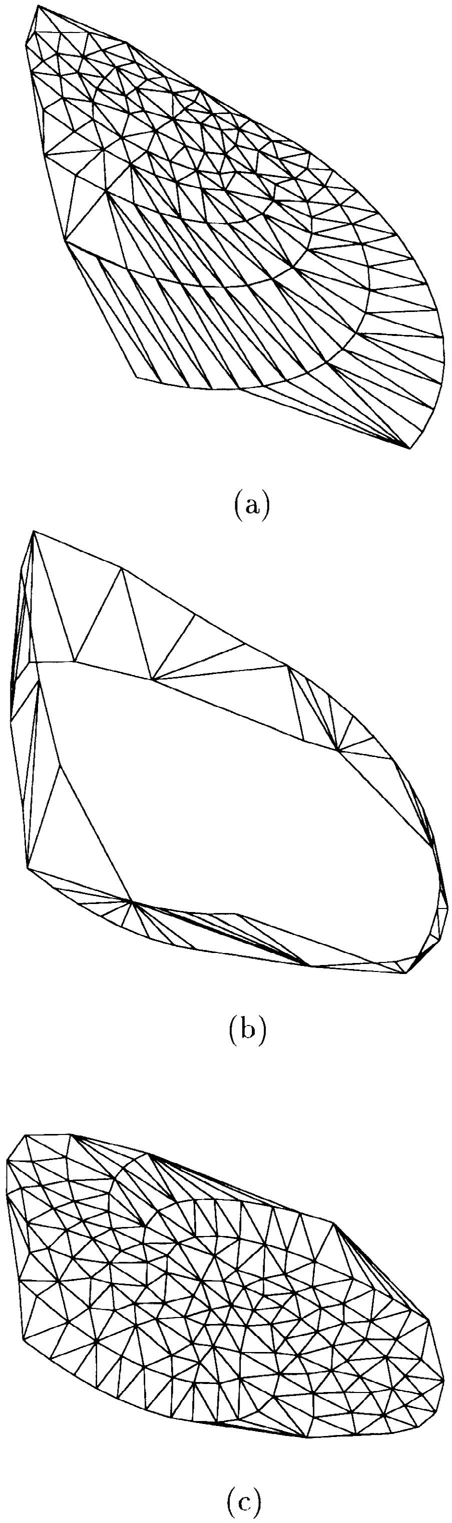


Figure 5.5: The boundaries of the sandwiched region in yz-projection: (a) top boundary; (b) side boundary; (c) bottom boundary.

As the trajectories enter a small neighborhood of the equilibrium, we use a linear feedback controller, such as the one discussed in [49], to stabilize the system at the equilibrium. Figure 5.6 shows the synthesized control reference trajectories originating at different initial displacements from the equilibrium: 1mm, 4mm, 4.5mm, and 5mm. The controller is able to recover from the first three initial points that are within the region. The last point is outside the region and thus uncontrollable; the current in the magnet can not build up fast enough to keep up with the ever-increasing airgap. The complete print-out of the synthesized reference trajectories showing the switching points is in Appendix B.

For example, the control law for the initial displacement of 1mm is specified as:

```
((time 0.) (switching-state #(140. 0. 220.)) (control 300.))
((time .0134) (switching-state #(163. 4.44 221.)) (control 140.))
((time .127) (switching-state #(139. .0789 202.)) (control *local-control*))
```

This control law specifies the time instance for each switching, the switching state, and the corresponding control value during the following time interval. The corresponding reference trajectory consists of trajectory segments delimited by these switching states.

5.3.3 Evaluating the control design

The synthesized global control law is a switching-mode one that changes the control parameter at the switching surface—the stable manifold of $V_i = 140$. It is able to bring trajectories originating from any states within the controllable region to a local neighborhood of the saddle.

The responses of the controller with respect to the four different initial displacements are shown in Figures 5.7 and 5.8. The vertical axis of each graph represents state variables x, y, and z as in the maglev model (5.1), one for each curve, and the horizontal one represents the time. For all the controllable initial displacements, the controller is able to bring the system back to the equilibrium with errors less than 0.2mm in displacement—a distance within the capture range of the linear feedback controller.

By exploring the state-space geometries of the maglev system, the Workbench is able to automatically determine the switching points for the global controller. The linear feedback controller can recover from only displacements of less than

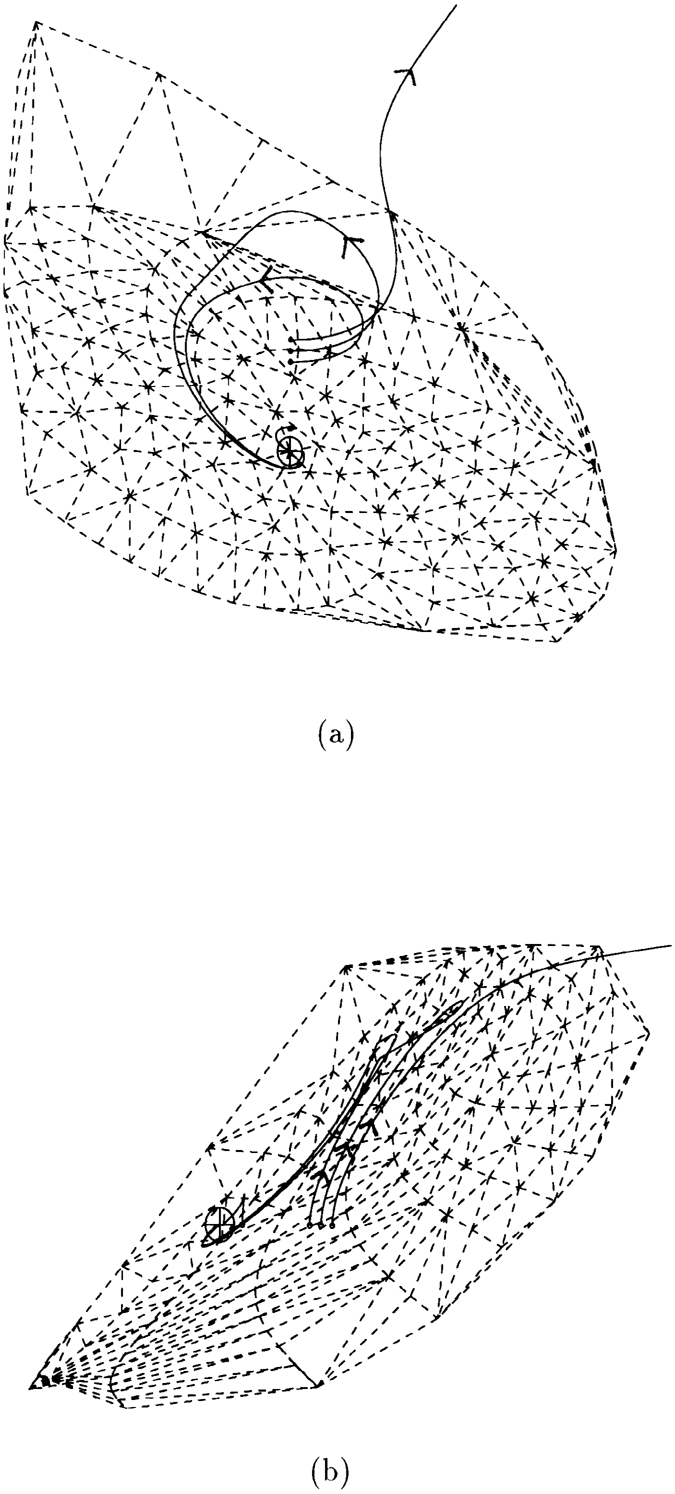


Figure 5.6: The synthesized control reference trajectories originating from four different initial states, together with the controllable region: (a) yz-projection; (b) zx-projection.

0.2mm. The global controller has significantly enlarged the operating region of the linear controller. With the geometric representation of the controllable region in state space, the Workbench precisely determined that the maximum recoverable displacement is 4.55mm. Our simulation geometrically explained the observation in [49] that the vehicle would fall off the rails with a displacement of 5mm or larger.

Many issues remain to be addressed in order to make the control design practical. Since our control law is designed with the nominal model of the maglev system, the effect of uncertainties in the model and of noise in the environment on the design needs to be studied in future research. The design can also be optimized with respect to response time.

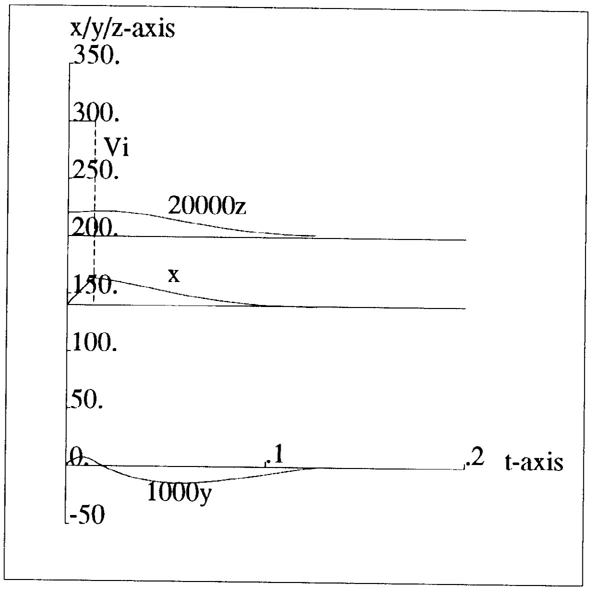
5.3.4 Visualizing the design

The synthesized controllable region is modeled with a polyhedral structure. This structure can be presented to engineers in a visual way. J. Choi has implemented a graphic rendering program for this purpose. Figure 5.9 shows a picture of the graphically rendered controllable region. The graphical presentation facilitates interactive, incremental modifications to the design in terms of the geometry. For example, one might want to directly manipulate the controllable region by tuning knobs corresponding to parameters. The geometric representation helps the designers to develop intuitions and to visually explore the effects of certain design choices.

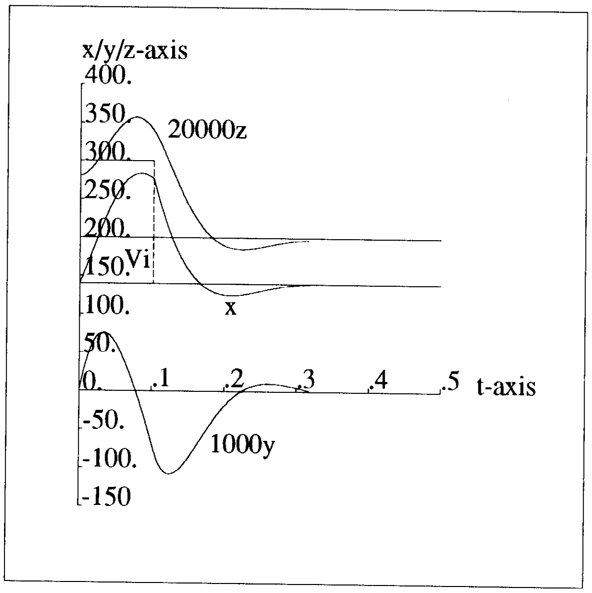
5.3.5 Implementation of the controller

The control design described above has been computationally simulated only. How will this design be implemented on the real system and used in real time?

The control law specifying the switching surfaces in state space can be compiled into a table. The control execution will be a table lookup and a geometric inequality test. At each step of the execution, the state of the system is sensed and checked against the switching surface. If the state is on the switching surface, the corresponding control value for the next time interval is read from the table and applied to the physical system. The implementation does not have to be very different from that for a dynamic programming one.

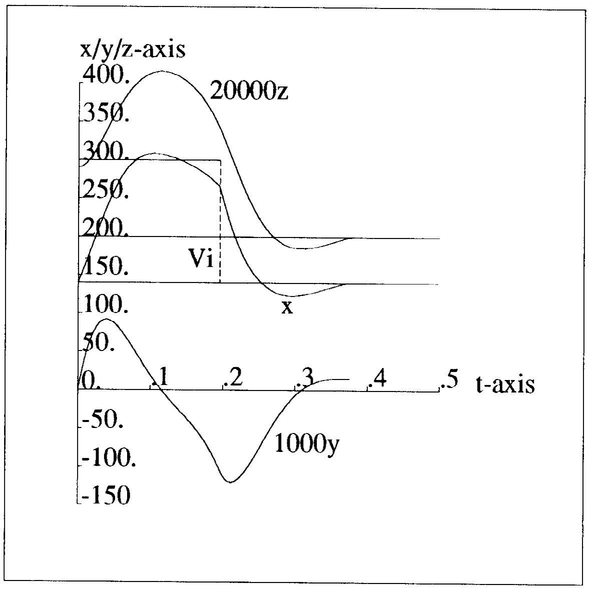


(a) $dz = 1mm$

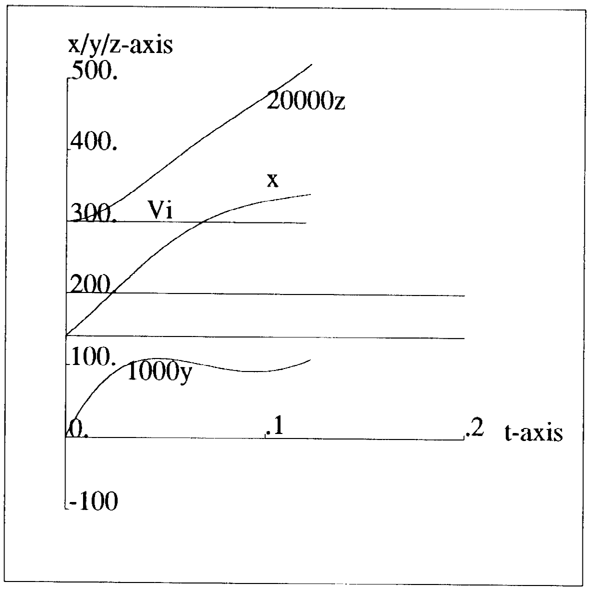


(b) $dz = 4mm$

Figure 5.7: Simulation of the nonlinear control design for different initial displacements. (to be continued)



(c) $dz = 4.5mm$



(d) $dz = 5mm$

Figure 5.8: (cont'd) Simulation of the nonlinear control design for different initial displacements.

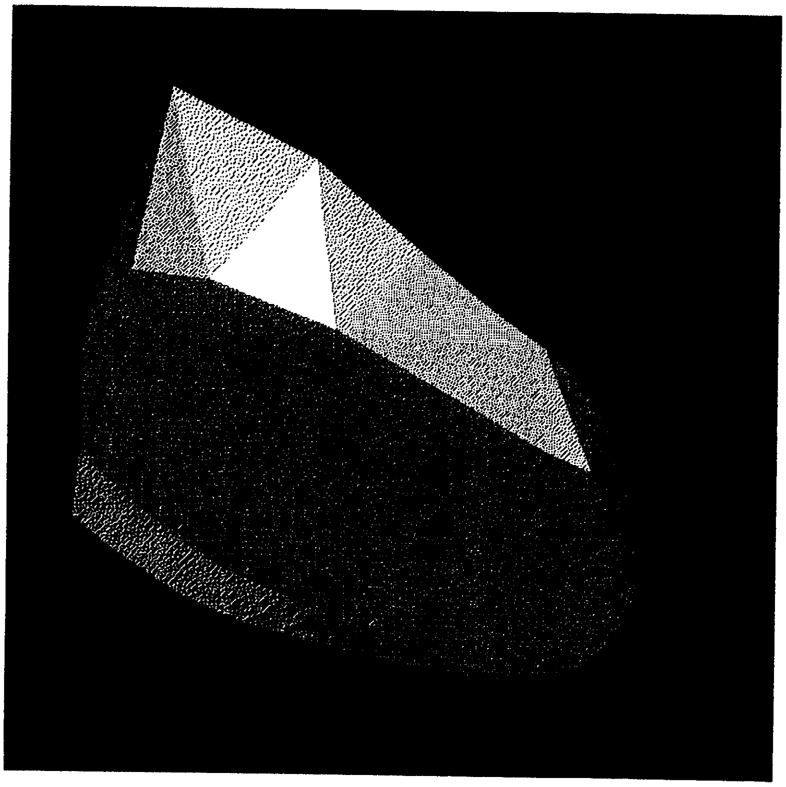


Figure 5.9: The controllable region for the maglev control design, rendered with the light source straight on. Hidden faces and lines are removed.

5.4 Summary of the Chapter

We have described the automatic design of a high-quality controller for stabilizing an EMS-mode maglev vehicle traveling above a guideway with the Workbench. The synthesized control reference trajectory consists of a sequence of trajectory segments, connected at intermediate points where the control voltage switches. At run-time, the controller will cause the system to track the reference trajectory and reactively correct local deviation from the desired trajectory. We have illustrated the state-space method for designing the global switching points of the nonlinear controller. The simulation showed that our design allows the maglev train to operate with much larger disturbances in the airgap than the classical linear feedback design does.

Chapter 6

Related Work

This chapter surveys relevant work in areas of simulation and analysis for complex dynamical systems and nonlinear control synthesis methods. Whenever possible, the Workbench will be compared with related systems elsewhere.

6.1 Qualitative Analysis of Dynamics

Qualitative analysis of dynamics is concerned with deriving symbolic descriptions and constraints for the dynamics of a physical system from simulations. The MAPS program of Chapter 3 falls into this category. Abelson and Sussman described a collection of computer programs that automatically analyze dynamical systems at the level of expert dynamicists [1, 2].

Yip has constructed a program, KAM, for automatically analyzing Hamiltonian systems with two degrees of freedom in two-dimensional phase sections [52]. The program uses techniques from computer vision to cluster trajectory points in phase sections and classifies phase portraits into meaningful categories. MAPS works in the domain of dissipative dynamical systems; it analyzes the systems in their continuous phase spaces regardless of their dimensions. In contrast, KAM applies to Hamiltonian maps on two-dimensional phase sections. MAPS constructs a simplicial representation for the phase-space qualitative features based on geometric pieces, as required by the task of control design. KAM uses a point-set representation instead.

Sacks' Poincaré program analyzes planar systems through a partition algorithm on phase spaces and a bifurcation analysis on one parameter [43]. The partition algorithm determines flow patterns in the plane using the properties of two-dimensional flows. These properties are based on the Jordan Curve Theorem and the fact that trajectories are one-dimensional curves that do not intersect. However, the special properties about two-dimensional flows do not generalize to higher dimensions. Recently, Sacks has been exploring the computation of two-dimensional manifolds in three dimensions using a set of trajectories, similar to that of MAPS. MAPS does not analyze bifurcations. MAPS differs from Sacks' partition algorithm in that MAPS is able to analyze phase spaces of any dimension, based on a general theoretical result on dynamical systems. In addition to its reasoning about the phase-space flow patterns, MAPS builds a workable representation in the form of a relational graph characterizing the spatial arrangement of phase-space objects. The representation contains information about the geometries of stability regions.

Eisenberg's Kineticist's Workbench is a program for simulating and explaining chemical reaction mechanisms [15]. It analyzes a chemical mechanism with numerical integrations and graph algorithms and interprets the simulation results in an episode data structure. The program does not use phase-space properties of chemical dynamical systems. Nishida et al. described a program for analyzing second-order nonlinear systems in phase space [34]. The program uses a flow grammar to specify possible patterns of solution curves in the phase plane and is restricted to two-dimensional flows.

Hsu developed a discrete, cell-based method called cell-to-cell mapping for approximating state spaces [21]. A continuous state space is discretized into regular cells forming a cell space. The associated map of a system becomes a cell-to-cell map which maps one cell to another cell. The cell-to-cell mapping method approximates the stability region of an attracting cell with a collection of cells that eventually map to that cell. MAPS does not take this uniform approach. Instead, it partitions a phase space into discrete regions according to trajectory flow patterns. The partition respects the geometry of the phase space. Because of the hierarchical (top-down) nature of Hsu's method, it can be integrated with MAPS in refining an approximation for a stability region.

Much of the difference between MAPS and the above cited work is due to the fact that the analysis MAPS performs is meant for control design. The representation in MAPS makes explicit the effects of certain design choices and is meaningful to control designers. For example, the geometries of stability regions are modeled in this representation; the sizes of the regions are directly linked to certain design parameters.

6.2 Engineering Stability Analysis

Another popular class of methods for engineering stability analysis is the so-called Liapunov-function based constructions for stability regions (domains of attraction). Our stability analysis of Chapter 3 does not use Liapunov functions; it is based on the stability-boundary characterization by Chang et al.

The class of Liapunov-function based approaches arises from the Liapunov theory of stability. It estimates domains of attraction by constructing suitable Liapunov functions. A Liapunov function is a family of closed surfaces such that a system trajectory remains in the region bounded by a surface after the trajectory enters. The Liapunov function is difficult to construct. Numerous algorithms have been proposed for this construction. Reference [17] gives a survey of these algorithms. Most of the algorithms assume special forms of the Liapunov functions and determine the unspecified coefficients in the functions. They usually result in conservative estimations. Margolis and Vogt [29] used the recursive method of Zubov for 2nd-order systems. The approach is restricted to systems with nonlinearities that possess Taylor series expansion and suffers from non-uniform convergence. Davison and Kurak [12] took the constrained minimization approach to fit a hyperellipse in the domain of attraction. Vannelli and Vidyasagar [51] used the concept of maximal Liapunov function to obtain a new partial differential equation. The partial differential equation provides a basis for an iterative procedure to compute the domains of attraction.

The complexity of these methods scales with the order of the systems and the order of the polynomials assumed for the Liapunov function. Manually computing the coefficients of the polynomials is very tedious. MACSYMA-like symbolic packages employing computer algebra techniques can automatically determine these coefficients [42]. Much work remains in the area of applying symbolic methods to the construction of the Liapunov functions.

6.3 Phase-Space Nonlinear Control

The gain scheduling and cellular control approaches use, to some extent, the phase-space representation of control systems to design nonlinear control laws. Like Phase

Space Navigator, they discretize a phase space into regions; but their decompositions are different and therefore bring to bear a different set of computational techniques.

In gain scheduling [45], the system is linearized around *a priori* selected operating points. A linear controller is designed for each operating point. Interpolation is used in regions between the operating points. Gain scheduling is an open-loop adaptation where parameters of a regulator change in a prespecified way. Autopilots and chemical process control are instances of gain scheduling. The method requires designers to decompose a phase space into locally linear regions. Phase Space Navigator differs from gain scheduling in that Phase Space Navigator automatically decomposes a phase space, even into regions that are global and nonlinear, and then synthesizes global reference trajectories using the knowledge of the phase-space decomposition.

We have compared the method of the Phase Space Navigator with the dynamic programming approach in Section 4.3.4. The cellular control method of Hsu [20] shares very much the same disadvantages as dynamic programming. The method is based on the cell state space concept used in the cell-to-cell mapping. It is similar to dynamic programming in discretizing the state space, but differs in that it also partitions the cost function space. The control law is synthesized while certain cost functions are minimized. The control value at each cell state is programmed into a table. Run-time control is to perform a table-lookup. This method requires an extensive search for control paths at very fine grain. The number of cells that the cellular method needs to visit could be very large even in a moderately-high-dimensional state space. The Phase Space Navigator, however, decomposes the phase space into a manageable collection of nonuniform subregions, the flow pipes, and searches for global paths in this collection.

Chaotic dynamical systems have extremely rich behaviors that can be tapped to design high-quality control systems. Recently, there have been several approaches to the controlling of chaos. Bradley developed a program called Perfect Moment that designs control paths for chaotic systems in phase space [8]. The program identifies phase-space chaotic features like strange attractors and searches for useful trajectory segments in a gridded phase space discretized into cells. Both Phase Space Navigator and Perfect Moment construct segmented control paths between the origin and destination states. Perfect Moment can be very useful in designing control paths for chaotic systems where the local structures of phase spaces are extremely complicated. On the other hand, Perfect Moment could not characterize the gross geometric features of a chaotic attractor, say the envelope, and use the global description of the attractor to guide the search; it relies entirely on the mapping of the grid of cells. Phase Space Navigator assumes there is no chaos. It searches in a space of flow pipes, as opposed to a grid of cells. It explicitly models the global geometries of phase space and trajectory flows. Perfect Moment's gridded search could be combined with Phase Space Navigator's global phase-space modeling to enhance the search for control trajectories.

The control design method developed in this thesis is not meant to replace other nonlinear control design techniques. To the contrary, the method complements the other techniques. The potential of the method in solving real problems can be fully realized when the method works together with other techniques. For example, feedback linearization is a well-developed nonlinear design method [45]. The method introduces a state feedback to a nonlinear system to cancel out the nonlinearity, thereby transforming the system into a linear one. The standard linear control design can be performed on the linear model. This linear design is then transformed back to work for the nonlinear system. Feedback linearization requires that the control input is continuous and the system has no zero dynamics. In cases when these conditions are met and a feedback linearizing control can be found, the feedback linearization could be an attractive alternative. On the other hand, Phase Space Navigator would be superior when, for example, the control takes discrete values as in switching-mode control.

6.4 Intelligent Control

The area of intelligent control, sometimes called knowledge-based control, has been very active recently. The knowledge-based approaches include direct expert control and indirect supervisory control [27]. The direct expert control encodes simple control laws into a rule base and invokes them at the run-time. The indirect approach influences the process by switching on and off controllers and updating control laws. For example, extreme alarm supervision switches on and off adaptors by monitoring the input signals and state variables. An intelligent control system usually has a hierarchical structure. The hierarchy ranks from high-level decision making to low-level PID[1] controls.

Phase Space Navigator can be regarded as one kind of intelligent control, if one wishes. It encodes deep domain knowledge of phase space and dynamics in the form of productions rules, algorithms, and other data structures. Different from the hierarchical structure in intelligent control, Phase Space Navigator consists of a global path planner and a local trajectory tracker; the two modules are integrated in a much tighter loop. This tight loop allows control decisions to be revised more naturally and less costly. For example, when the local tracker cannot correct trajectory deviation with local means, it calls upon the global planner to synthesize a new reference trajectory. An "intelligent controller" would have to pass the request and transform the decision back and forth through the control hierarchy.

---

[1] PID stands for proportion, integral, and derivative.

Chapter 7

Conclusions

7.1 Thesis Revisited

This thesis has developed a flow-pipe based phase-space method for designing nonlinear controllers and a qualitative representation for encoding dynamics. It has formulated the task of control design and analysis as a computational one in which computations and reasoning about dynamics are essential. It has constructed a Workbench comprising the programs MAPS and Phase Space Navigator to assist control engineers and to automate a significant portion of the control design task. The programs have been applied to the design of a nonlinear controller for a magnetic levitation vehicle.

MAPS analyzes the phase-space structure of a system in terms of geometric features. It produces a high-level description of the phase space that can be used to focus and prune the search for control paths. Because of its human accessible aspect, the description provides meaningful information to control engineers in their design process. Phase Space Navigator automatically synthesizes control reference trajectories using the phase-space description as a "map". The "map" models the phase space with trajectory flow pipes and provides an efficient representation for the search. We described the algorithms for finite-valued control synthesis and mentioned caveats in generalizing them to continuous, multiple control parameter spaces in Section 4.3.4. When the computations are tractable, the method has great leverage for grossly nonlinear systems.

The Workbench embodies domain knowledge from control engineering and dynamical systems theory. It understands concepts like asymptotic and transient behaviors, stability regions, reachable sets, convergence, overshooting, etc. The

Workbench organizes the modules of analysis, design, and graphics presentation around the control task and encourages incremental changes to the Workbench, for example, incorporating programs tailored to particular applications and encoding knowledge of specific domains. Suppose we want to use the Workbench to design electric power control systems. In addition to their differential equation models, the power systems have special structural properties and model formulations that can be exploited to reduce design complexity and improve design quality. Special program fragments exploiting these properties can be integrated into the Workbench. The integration is made possible and easier by the underlying Scheme implementation as a substrate. The Scheme programming language facilitates composition and abstraction of procedures.

7.2 Problems and Future Work

We examine the situations in which the Workbench does not work, analyze the reasons for its breakdown, and suggest further enhancements to the Workbench[1] and future areas where much exciting work remains to be done.

MAPS performs stability and flow analysis for a continuous dynamical system in a bounding box. It assumes that the system contains no chaotic attractors and fractal boundaries. MAPS can also analyze steady-state trajectories for discontinuous systems; in fact, it has successfully analyzed the limit-cycle behaviors for a switching power regulator, a series resonant converter with clamped tank capacitor voltage, under various operating conditions [25].

To extend the stability and flow analysis of MAPS to work for systems with discontinuities is difficult. One reason is that the theoretical basis of the stability analysis breaks down. Theoretical results for stability characterizations of discontinuous systems are scarce [16]. Computational simulation and modeling techniques, combined with *ad hoc* treatments for limit cycles, could provide an attractive alternative for analyzing discontinuous systems.

Chaotic attractors are difficult to characterize. The boundaries and the attractors themselves can be fractal. Their fractal boundaries are typically caused by the heteroclinic intersections of stable and unstable manifolds. The fine, convoluted structures pose great problems for numerical algorithms attempting to determine them. The stability and robustness of the computation is the main issue in this case.

---

[1] Chapter 3 has discussed several extensions to the MAPS program.

The Workbench could be extended to characterize gross geometric features of chaotic attractors. It could augment purely numerical measures with some invariant, symbolic characterizations. Many chaotic attractors, for example the Rössler band [5], are constructed with a stretching-and-folding operation. The Workbench could compute the Birkhoff signature encoding the heteroclinic intersections of stable and unstable manifolds. This symbolic sequence plus the stretching-and-folding operation can help understand the fractal structures. The Workbench could also calibrate dimensions of the attractors and determine Liapunov exponents. The combination of geometric descriptions, symbolic constraints, and numerical measures can better characterize the fine structures of chaotic attractors.

The Workbench has adopted the Euclidean metric in n-dimensional Euclidean spaces. However, a different metric, the one on surfaces of manifolds rather than on their embedding spaces, could also be used. This new metric would be able to more accurately model the folding of a manifold. The information about the geometric properties of the manifold can be used to improve the geometric modeling algorithm discussed in Chapter 3. In addition to the metric information, the Workbench could use certain topological information about the space in planning control. We have implemented a program for computing the homology group of a triangulated manifold based on a reduction algorithm [33]. It will be interesting to see how we could use this topological property of phase portraits to design control paths. The Workbench could also be extended to work for other types of spaces such as projective spaces and tori.

The design capability of Phase Space Navigator can be enhanced with other optimization techniques. For example, the variational method can be used in optimizing the local control trajectories after the global path is established. The cost of the optimization at the stage of local trajectory generation is expected to be low.

Future research should address the robustness of the phase-space control design regarding for example model order uncertainties or unmodeled dynamics. The technique of "thickened trajectory" discussed in Section 4.3 models only measurement uncertainties or noise. The structural uncertainties of the model could change the topological structure of a phase space. The effect of such uncertainties on the phase-space reference trajectories remains to be quantified. The thesis has not addressed model building for dynamical systems. The Workbench could interface with a modeling program that allows the Workbench to deal with a wider class of systems. The modeling program should formulate the most appropriate model for the Workbench.

The current implementation of the Workbench is expected to work well for moderately low-dimensional systems. However, "the curse of dimensionality" makes the computations intractable for high-dimensional systems. This is somewhat a paradox: high-dimensional, highly-nonlinear systems exhibit richer flow patterns in phase space and provide more design opportunities for the phase-space method. However, such advantage is at the expense of more computations. To extend the applicable range of the method, the efficiency of the geometric algorithms needs to be greatly improved.

7.3 Broad Implications

The work reported in this thesis has broad implications:

The foremost is its demonstration that controllers for complex systems can be automatically designed. The work showed that programs can visualize qualitative behaviors in phase space; programs can plan control trajectories and steer systems; and the phase-space geometric modeling provides a "map" for navigating systems in phase space.

Second, novel computational representation and reasoning mechanisms can be developed in the context of automating challenging engineering tasks. The dynamics of nonlinear systems is difficult to describe and manipulate. The qualitative representation developed in this thesis provides a way to computationally describe the qualitative aspects of the dynamics. With this representation, the difficult control design is translated into a computational task: the flow-pipe based mechanism manipulates a system's natural dynamics and synthesizes the desired dynamics for the system.

Third, the domain knowledge and techniques from symbolic, numerical, and geometric computing have been proven essential. In order to fully exploit the dynamics and to build programs to imitate human control designers, the Workbench uses whatever knowledge and techniques that are necessary: geometric theory of dynamical systems, control theory, and techniques from artificial intelligence, vision, computational geometry, numerical analysis, and graph search.

Fourth, the complexities of the control design task necessitate the need for automatic modeling and analysis tools. Autonomous programs like those in the Workbench have extended the capabilities of the "eyes" and "hands" of control engineers in seeing and manipulating objects. They have enlarged the design space engineers can explore. In certain cases, programs can even outperform human designers.

Fifth, the phase-space trajectory planning has the potential of becoming a powerful new paradigm for nonlinear control synthesis and a complementary alternative to other control design techniques.

Finally, the potential applications of the Workbench are numerous: (1) Large-scale power systems are difficult to analyze and control. A power system needs to be analyzed in terms of its stability margin. It needs to be brought back to the original equilibrium after a power outage. The Workbench could be enhanced to provide geometric means for the stability analysis and possibly for the control trajectory design. (2) Vehicles like the maglev trains and automobiles usually use passive suspension to dampen disturbances caused by road irregularities and wind loads. It is conceivable that such disturbances could be compensated with active actuation. This technology is called active suspension. In the maglev example, it is possible to characterize the guideway irregularities; thus active suspension could be an attractive candidate. The Workbench can help in designing such a secondary suspension for the maglev vehicle that will complement the primary suspension (magnetic levitation) designed in Chapter 5. (3) Large flexible space structures use thin beams as building blocks in structures like the truss. These beams work in nonlinear regions. The Workbench can be used to study the buckling of the beams and explore control schemes to strengthen them. There are many more areas where the Workbench will find itself useful.

7.4 Towards A Language for Computational Control Design

We outline a computational language with which one can describe aspects of the control design. Although the language is in a very preliminary form, elucidating basic constructs and properties of the language helps capture the informal design procedures and serves as a stepping stone for formalizing the knowledge of control engineering.

Professional control engineers make use of implicit working knowledge such as the phase-plane method and work at different design levels towards the goal. By expressing the informal design knowledge explicitly in computational terms, we are able to build programs to manipulate the knowledge, mechanize certain laborious tasks, communicate the knowledge more effectively to other designers, and improve education in control engineering. In addition, the language provides means for building design abstractions, for capturing design rationales, and for addressing computational complexities of the design.

The design language consists of a set of primitives and operations on the primitives. The primitives are phase portraits segmented into flow pipes. We call these segmented portraits the flow-pipe portraits. The operations combine these flow-pipe portraits to form new ones. An operator also takes conditional choice as an argument. In composing two flow-pipe portraits, the operator intersects the flow pipes in the portraits and glues them together at intersections. The choice information specifies for flow pipes which way to go at each intersection. The semantics of the language is dictated by the underlying dynamics of the system and the control task. For example, an intersection of two flow pipes is specified as a geometric intersection of their polyhedral approximations.

We express the phase-space control design in a high-level form with this language: the control design is the composition of flow-pipe portraits. The optimization of the design is equivalent to the minimization of certain measure in the composition process. When an optimization criterion is specified, the composition process constructs the flow-pipe portrait containing the optimal control path.

7.5 The Big Picture

The domain of control synthesis is interesting to study for several reasons. First, the computational codification of engineering knowledge such as the knowledge of control design enables us to distill and communicate principles of engineering design in a more explicit way. Second, practical control systems are extremely useful but are often difficult to design. Intelligent design aids for control engineers like the Control Engineer's Workbench can improve the design quality and increase the productivity.

I am convinced that high-performance computational techniques have great potentials in revolutionizing traditional engineering computation and design, and research investigating challenging real-world problems serves as a driving force for the development of computer science and artificial intelligence. This thesis serves as an instance for demonstrating my conviction.

This work has primarily focused on the automatic synthesis of global nonlinear controllers and has not addressed the important issues of modeling, sensing, estimation, and high-quality linear controller design. The exploration and development of a collection of new methods tackling these issues in all dimensions can revolutionize the synthesis and analysis of high-performance nonlinear control systems.

The research described in this thesis actively exploits new synergies among artificial intelligence, computer science, applied mathematics, dynamical system theory, and control engineering and develops appropriate computational techniques to tackle real problems in engineering. The Control Engineer's Workbench is a prototype of a new class of intelligent computational tools that combine numerical and symbolic computations with AI reasoning techniques and automatically model, analyze, and design complex physical systems. These tools, when used as human aids, will greatly amplify and extend our capabilities in seeing and manipulating the world around us.

Appendix A

Maglev Application: Equilibria

The equilibria for $V_i = 140$ and $V_i = 300$:

```
; Vi = 140
((#(140. 0. 200.)                ; equilibrium point #(x y z)
  (-17.004+22.963i               ; eigenvalue 1
   -17.004-22.963i               ; eigenvalue 2
   24.007)                       ; eigenvalue 3
  (#(.23604 .97174 0)            ; eigenvector 1
   #(.51331 -.55588 .65384)      ; eigenvector 2
   #(.30157 .73255 .61027))      ; eigenvector 3
  saddle))                       ; stability type
```

```
; Vi = 300
((#(300. 0. 428.57)              ; equilibrium point #(x y z)
  (-21.411+21.394i               ; eigenvalue 1
   -21.411-21.394i               ; eigenvalue 2
   21.394)                       ; eigenvalue 3
  (#(.54819 .83636 0)            ; eigenvector 1
   #(.43085 -.65947 .616)        ; eigenvector 2
   #(.23229 .71052 .66423))      ; eigenvector 3
  saddle))                       ; stability type
```

Appendix B

Maglev Application: Synthesized Reference Trajectories

This appendix contains the initial, switching, and final points of the synthesized reference trajectories starting from four different initial states, as shown in Figure 5.6. Each point in the print-out is a vector of (t, x, y, z).

```
; trajectory 1
;   init-pt: #(0. 140. 0. 220.)
((time 0.) (switching-state #(140. 0. 220.))
 (control 300.))
((time .013441) (switching-state #(162.98 4.4416 221.42))
 (control 140.))
((time .12704) (switching-state #(139.39 .078872 201.75))
 (control *local-control*))

; trajectory 2
;   init-pt: #(0. 140. 0. 280.)
((time 0.) (switching-state #(140. 0. 280.))
 (control 300.))
((time .10156) (switching-state #(276. -74.214 338.79))
 (control 140.))
((time .32041) (switching-state #(139.26 -.19212 196.51))
 (control *local-control*))
```

```
; trajectory 3
;   init-pt: #(0. 140. 0. 290.)
((time 0.) (switching-state #(140. 0. 290.))
 (control 300.))
((time .19536) (switching-state #(264.69 -105.03 340.68))
 (control 140.))
((time .42865) (switching-state #(142.4 3.2661 202.69))
 (control *local-control*))

; trajectory 4
;   init-pt: #(0. 140. 0. 300.)
#(0. 140. 0. 300.) is outside controllable region
```

Bibliography

[1] H. Abelson and G. J. Sussman, "The Dynamicist's Workbench I: Automatic Preparation of Numerical Experiments." In: *Symbolic Computation: Applications to Scientific Computing.* R. Grossman (ed.), SIAM, Philadelphia, PA, 1989.

[2] H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G.J. Sussman, J. Wisdom, and K. Yip, "Intelligence in Scientific Computing." *CACM*, 32(5), May 1989.

[3] H. Abelson, "Bifurcation Interpreter: A step towards the automatic analysis of dynamical systems." *Int. J. of Computers and Mathematics with Applications*, 20(8), June 1990.

[4] H. Abelson, A. Berlin, J. Katzenelson, W. McAllister, G. Rozas, and G.J. Sussman, "The Supercomputer Toolkit and its Applications," *Proc. of the Fifth Jerusalem Conference on Information Technology*, Oct. 1990.

[5] R. H. Abraham and C. D. Shaw, *Dynamics—The Geometry of Behavior.* Aerial Press, Santa Cruz, CA, 1988.

[6] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1962.

[7] Jean-Daniel Boissonnat, "Geometric Structures for Three-Dimensional Shape Representation." *ACM Trans. on Graphics*, 3(4), October 1984.

[8] E. Bradley, "Control Algorithms for Chaotic Systems." First European Conference on Algebraic Computing in Control, 1991.

[9] E. Bradley and F. Zhao, "Phase-Space Control System Design." To appear in *IEEE Control Systems Magazine*, 1992.

[10] H. Chiang, M. W. Hirsh, and F. F. Wu, "Stability Regions of Nonlinear Autonomous Dynamical Systems." *IEEE Trans. Automatic Control*, 33(1), Jan 1988.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1990.

[12] E. J. Davison and E. M. Kurak, "A computational Method for Determining Quadratic Lyapunov Functions for Non-linear Systems." *Automatica*, 1:627-636, 1971.

[13] R. C. Dorf, *Modern Control Systems.* Addison-Wesley, MA, 1989.

[14] T. R. Eastham, "Maglev: A Realistic Option for the Nineties." In *Magnetic Levitation Technology for Advanced Transit Systems,* A collection of papers presented at SAE Conference and Exposition on Future Transportation Technology, Society of Automotive Engineers, 1989.

[15] M. A. Eisenberg, "The Kineticist's Workbench: Combining Symbolic and Numerical Methods in the Simulation of Chemical Reaction Mechanisms." *MIT AI-TR*-1306, June 1991.

[16] A. F. Filippov, "Differential Equations with Discontinuous Right-Hand Side." Americal Mathematical Soc. Translations, Series 2, Vol 42, pp 199-231, 1964.

[17] R. Genesio, M. Tartaglia, and A. Vicino, "On the Estimation of Asymptotic Stability Regions: State of the Art and New Proposals." *IEEE Trans. Automatic Control*, AC-30(8):747-755, August 1985.

[18] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, 1983.

[19] M. W. Hirsh and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra.* Academic Press, NY, 1974.

[20] C. S. Hsu, "A Discrete Method of Optimal Control Based upon the Cell State Space Concept." *J. Optimization Theory and Applications,* 46(4):547-569, August 1985.

[21] C. S. Hsu, *Cell-to-Cell Mapping.* Springer-Verlag, New York, 1987.

[22] C. S. Hsu and R. S. Guttalu, "An Unraveling Algorithm for Global Analysis of Dynamical Systems: An Application of Cell-to-cell Mappings." *J. Applied Mechanics,* 47:940-948, December 1980.

[23] U. Itkis, *Control Systems of Variable Structure.* John Wiley, New York, 1976.

[24] A. Isidori, *Nonlinear Control: An Introduction.* Springer-Verlag, New York, 1985.

[25] B. S. Jacobson and R. A. DiPerna, "Series Resonant Converter with Clamped Tank Capacitor Voltage." IEEE Applied Power Electronics Conference, 1990.

[26] R. E. Kalman, "Phase-plane Analysis of Automatic Control Systems with Nonlinear Gain Elements." *Trans. of AIEE,* 73(2):383-390, Jan. 1954.

[27] A. J. Krijgsman, et al., "Knowledge-based Control." *Proc. 27th conf. Decision and Control,* Austin, TX, December 1988.

[28] The Maglev Technology Advisory Committee, "Benefits of Magnetically Levitated High-Speed Transportation for the United States." Volume 1, Executive Report for the United States Senate Committee on Environment and Public Works. Published by Grumman Corp., Bethpage, NY, June 1989.

[29] S. G. Margolis and W. G. Vogt, "Control Engineering Applications of V. I. Zubov's Construction Procedure for Lyapunov Functions." *IEEE Trans. Automatic Control,* AC-8:104-113, April 1963.

[30] The MathWorks, SIMULAB*, a program for simulating dynamic systems, a user's guide.* The MathWorks, Inc., 1990 and MATLAB *User's Guide.* The MathWorks, Inc., 1989.

[31] F. C. Moon, *Chaotic Vibrations.* John Wiley & Sons, New York, 1987.

[32] F. C. Moon and P. J. Holmes, "A Magnetoelastic Strange Attractor." *J. Sound Vib.*, 65(2), 1979.

[33] James R. Munkres, *Elements of Algebraic Topology.* Addison-Wesley Publishing Company, Reading, MA, 1984.

[34] T. Nishida, K. Mizutani, A. Kubota, and S. Doshita, "Automated Phase Portrait Analysis by Integrating Qualitative and Quantitative Analysis." *Proc. of AAAI-91*, July 1991.

[35] K. Ogata, *Modern Control Engineering.* Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[36] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a Time Series." *Physical Review Letters,* 45:712-716, 1980.

[37] O. Palacios-Velez and B. C. Renaud, "A Dynamic Hierarchical Algorithm for Computing Delaunay Triangulations and Other Closest-Point Problems." *ACM Trans. on Mathematical Software,* 16(3):275-292, September 1990.

[38] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems.* Springer-Verlag, New York, 1989.

[39] H. Poincaré, "Mémoire sur les courbes définies par une équation différentielle." *Journal de Math.*, 7:375, 1881.

[40] F. P. Preparata and M. I. Shamos, *Computational Geometry.* Springer-Verlag, New York, 1985.

[41] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, Numerical Recipes: the art of scientific computing. Cambridge University Press, Cambridge, 1988.

[42] R. H. Rand, *Computer Algebra in Applied Mathematics: An introduction to* MACSYMA. Pitman, Boston, 1984.

[43] E. Sacks, "Automatic Analysis of One-Parameter Planar Ordinary Differential Equations by Intelligent Numeric Simulation." *CS-TR-244-90*, Princeton University, Jan 1990.

[44] Y. Sakawa and Y. Shindo, "Optimal Control of Container Cranes." *Automatica*, 18(3), 1982.

[45] J. E. Slotine and W. Li, *Applied Nonlinear Control.* Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[46] S. Smale, "Differentiable Dynamical Systems." *Bulletin of the AMS*, 73, 1967.